The Use of System Level Synthesis Tools in Education

Allen Dutoit, Janaki Akella, Daniel Siewiorek

EDRC 18-26-92

The Use of System Level Synthesis Tools in Education

Allen Dutoit
adlh@edrc.cmu.edu

Janaki Akella
jak@edrc.cmu.edu

Prof. Daniel P. Siewiorek
dps@cs.cmu.edu

December 1991

Abstract*

This document describes the use of system level synthesis tools in a post college, industrial course taught over a twelve week period during the summer of 1991. The course was offered by the Engineering Design Research Center and the Carnegie Bosch Institute, both at Carnegie Mellon University. The course was intended for technical and high-level managers interested in improving their knowledge about design theory and software tools to support the design process.

The course was divided into a lecture part and a project part. During the project, the participants were to design, manufacture and test a small computer for displaying blueprints. This report focuses on the requirements, the preparation and the implementation of the project part of such a course.

* This work has been supported by the Engineering Design Research Center, a NSF Engineering Research Center.

Table of Content

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The main purpose of this report is to record and to transmit information. The Bosch course 91 was the first of its kind at Carnegie Mellon University, with respect to goals, the number and the background of participants, the number of people involved in preparation, and the amount of resources consumed. Most of the people who worked for this course had little experience in teaching an industrial course and acquired on-the-job training. Our goal is to record the experience learned during this course and to transmit this knowledge to the people who want to organize a similar course in the future.

A second goal of this document is to initiate a discussion on the relationship of technology transfer courses and research at EDRC. Such courses can be valuable for obtaining early feedback on the results of research and providing comments on how such results can be applied in an industrial context. However taking advantage of this feedback is far from being a trivial task.

The authors of this document are respectively the student who was responsible for the tool preparation, the student responsible for the design of the artifact, and the faculty member in charge of the project part of the course.

1.2 Structure

Section 2 defines the terms used in this report and gives some background information on the course.

Section 3, 4 and 5 focus respectively on the design process, the tool preparation and the teaching aspects of the project. These sections contain data on the resources required and the problems encountered during the preparation of each of these aspects of the project. They also propose alternates for solving and preventing the identified problems. The purpose of these sections is twofold. First to provide enough data to allow an estimation of the time and resources needed to prepare a similar course. Second, to simplify an estimation of contingency resources.

Section 6 contains remarks on the overall project preparation. We conclude in Section 7 by including some thoughts on the relationship of such courses to research.

2 Background

2.1 Goal of the course and the project

As stated in the brochure distributed to the participants of the course[1], the goal of the program was:

- "to supplement the participants' personal knowledge and experience in engineering design with that of leading scientists,
- to introduce state-of-the-art tools and methods, and
- to help the participants explore new approaches to the design process itself and to the management and organization of design teams."

A significant part of the program was a project performed by the participants in small groups (six participants). The purpose of the project was to provide the participants with an environment for experimenting with various approaches to design and different team organizations. The participants designed a prototype of a small portable computer intended to display blueprints for construction workers (described in Sections 2.4.2 and 3.3). Both commercial software packages and software tools developed at EDRC were used.

A secondary goal was to test individual tools developed at EDRC in a larger scale environment, and to accelerate the integration of tools from different projects into a single design environment.

2.2 Terms used in this document

The faculty and students who prepared and executed the course are referred as the *project team*. The subset of the project team who designed and manufactured the prototype is referred as the *project design team*. Similarly, the students and the tool providers who prepared the tools for the course are referred as the *tool preparation team*.

The industrials who attended the course are referred as the participants. The participants were also divided into teams we call *participant design teams*.

Tool refers to any in-house or commercial piece of software which was used during any of the design processes. *Tool integration* refers to the effort of bringing the tools into a single software environment, in which some of the interactions between tools are hidden from the user.

The term *artifact* refers to the result of any of the design processes. The artifact produced by the project design team is often referred as the *golden solution*.

1. Carnegie Bosch Institute, Advanced Programs for Technical Managers, Program in Engineering Design, May 13 - August 23, 1991

2.3 Participants

The course targeted technical and high-level managers seeking to improve their knowledge about design theory and practice. Twenty six participants from North American and European industries attended the course. Figure 1 classifies the participants by their staff sizes. Each category corresponds to a staff size, and not to a job title.

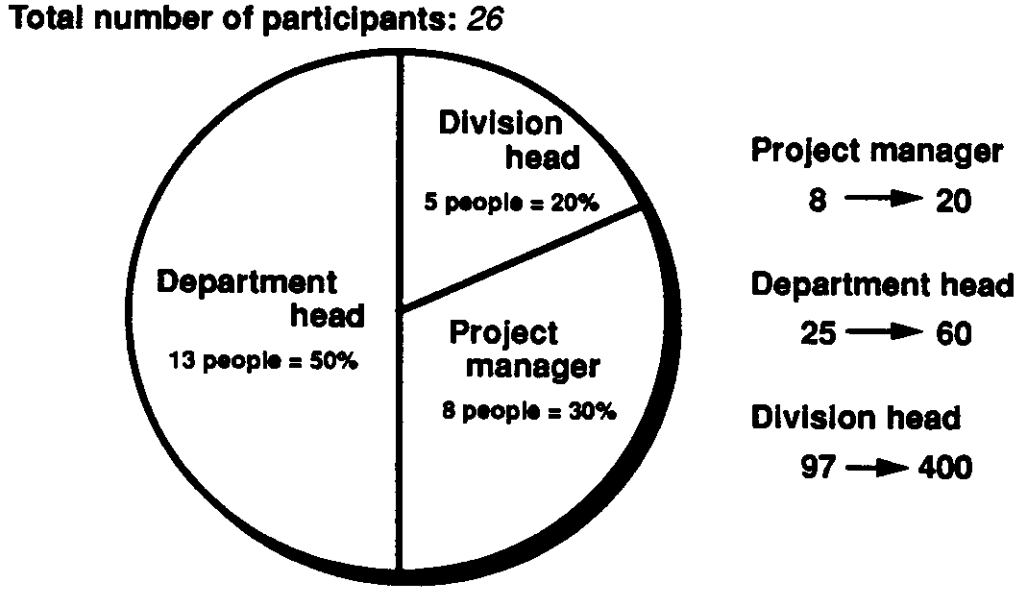


Figure 1 Participants by staff size

Figure 2 describes the academic background of the participants, i.e. the engineering field they studied last in college. In most of the cases, this also represents the field which they are currently working in.

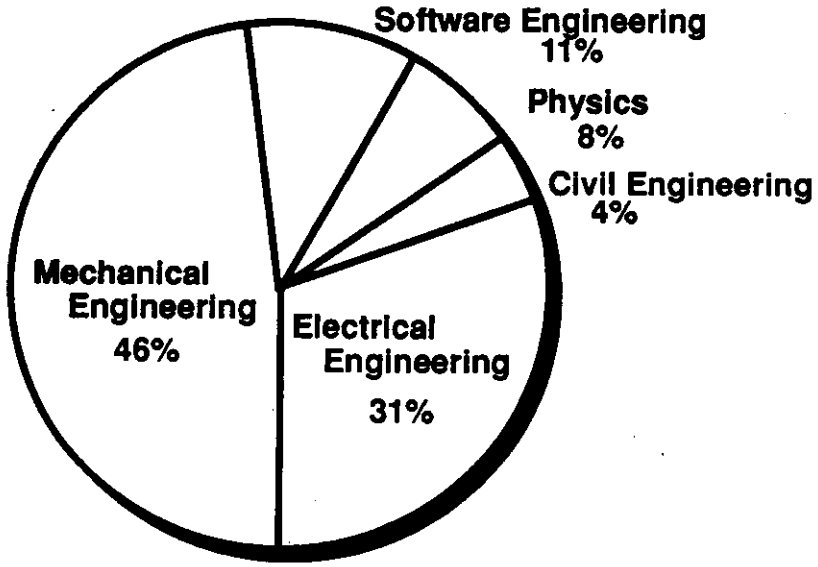


Figure 2 Participants by academic background

Figure 3 shows the participants by age. Except for people working in software, we found that the level of computer skills of the participants varied inversely with their age. One purpose of some participants at the high end of the scale for attending the course was to improve their specific knowledge about the current capabilities of CAD tools.

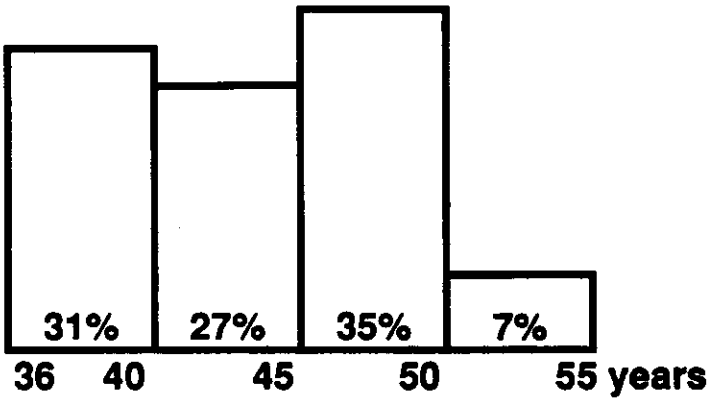


Figure 3 Participants by age

Figure 4 classifies the participants by activity. Many participants were managing design and development teams. Surprisingly, few of them were directly connected to CAD.

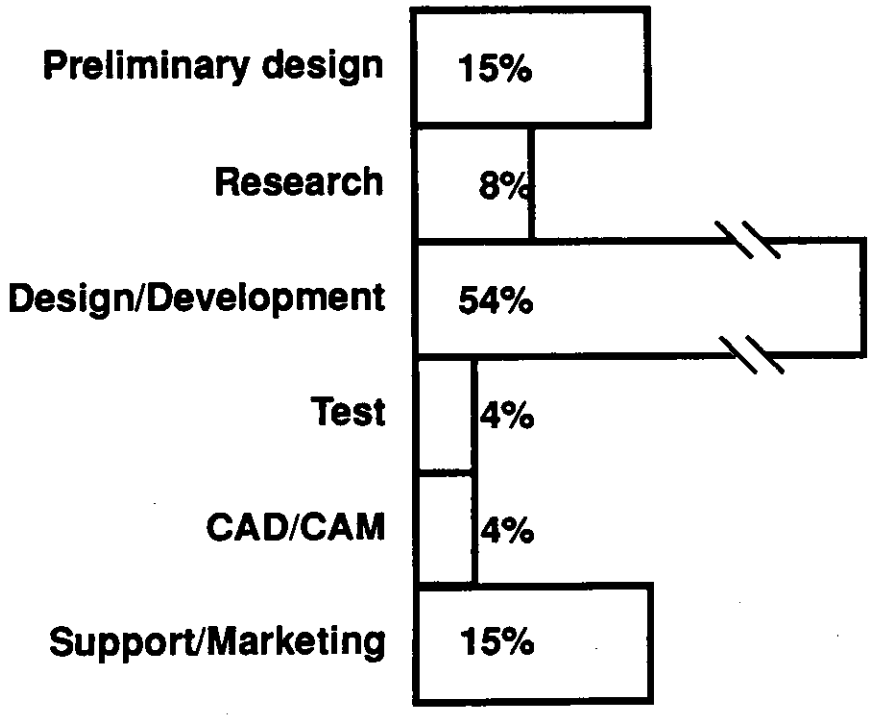


Figure 4 Participants by task

2.4 Target project

2.4.1 Philosophy

The intent of the project was to lead the participants through a complete design and manufacture process in order to give them an environment to experiment with the ideas and methods presented during the lecture part of the course. The course lectures were given six mornings per week, the project took place five afternoons per week. The course lasted twelve weeks, the project ten.

The project design team started to design the artifact prior to the beginning of the course. The design process consisted of five steps:

- product concept
- electronic specification
- electronic design
- housing specification
- housing design

The participants were divided into four teams of six members. The team assignment was done according to the background of each participant. Each team was to design a similar artifact from the same set of requirements. The structure of their design processes was similar to the project team design process.

Figure 5 shows the interactions between the design processes of the project team and the participant teams. At each step, the participant teams solved a problem. The four different solutions were then critiqued. The solution of the project design team (called the golden solution) was then used as the starting point of the next step of the participants' design processes. In a few cases (e.g. the housing specification), we included some aspects of the participants' solution into the golden solution.

The reason for bringing all the teams back to the same starting point for the next step, was to not spend an enormous amount of personpower in supervising diverging projects. The design process steps occurred sequentially, again to conserve instruction effort.

Once the design was completed, the participants were to participate in the manufacture and test of the golden solution. Since the teams were refocused at each step, the manufactured artifact would be similar to the one they would have designed.

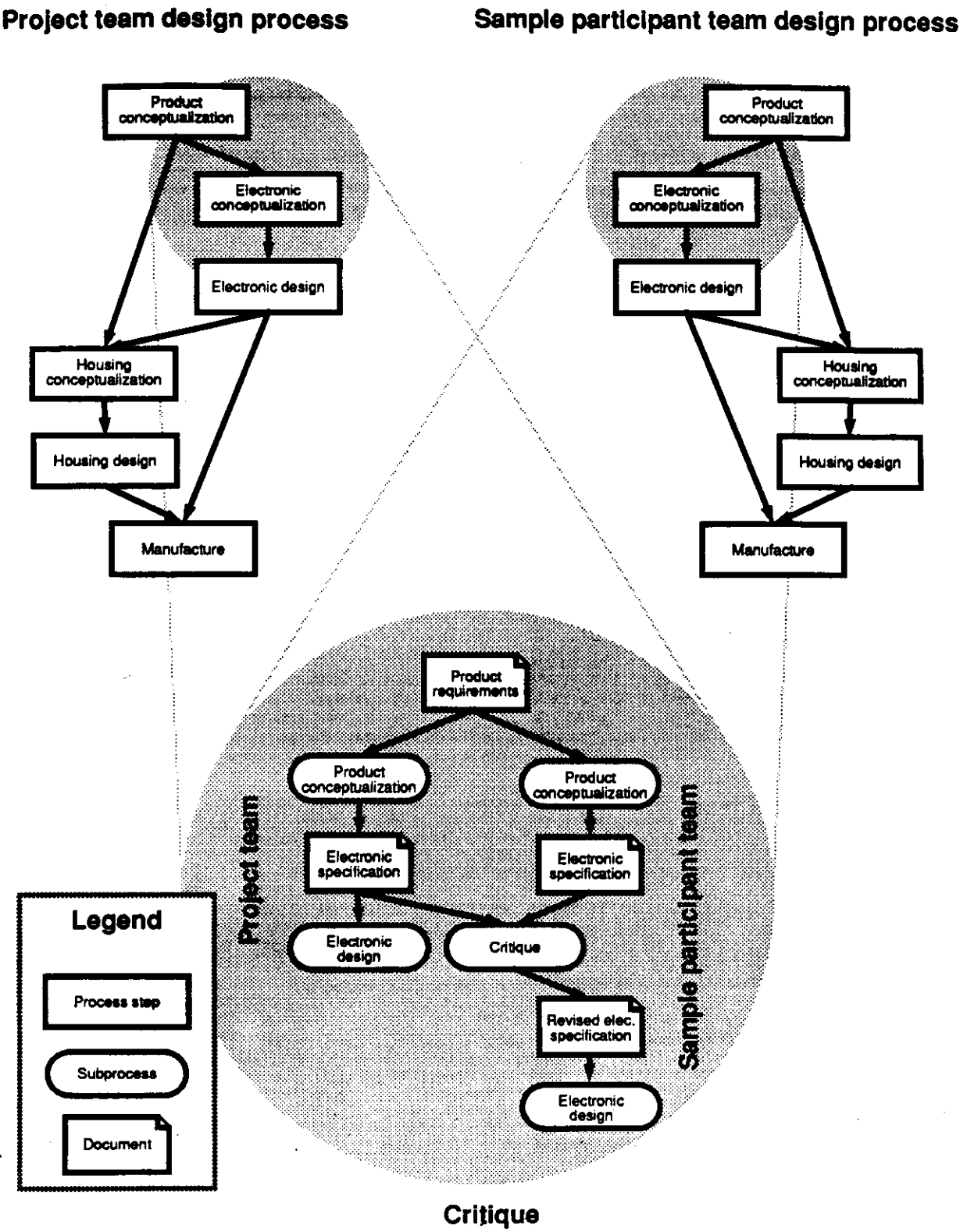


Figure 5 Interactions between design processes

2.4.2 Artifact

The artifact the participants were to design was a small computer for displaying blueprints. The initial specification was to display maps in an automobile. Shortly after the beginning of the course, the artifact was changed from a map display to a construction blueprint display. Some of the participants were involved in the design of an automobile map reader in their company. They did not want to work on the design of such an artifact in the context of a course, claiming that any idea they had on this subject was the intellectual property of the company they were working for. However, the requirements of the electronic portion of the design did not change.

The computer was specified to be small enough to carry in the field. The functionality of the computer were to include: switching between two different blueprints, scrolling, and zooming. The user interface of the computer had to be simple enough to be operated safely while walking.

Another requirement of the artifact was that it should use a Private Eye as the display. The Private Eye is a 3/4 by 3/4 inch screen, mounted on a headset. Held close to the eye, it gives the illusion of a regular 12 inch monitor screen floating in the air. The Private Eye has similar properties as bifocal glasses: the user can either look at the outside world or at the screen with minimum eye focusing.

The Private Eye was not included in the first specification, but introduced as an 'arbitrary and managerial' decision in the second phase, in order not to constrain the design from the beginning.

2.4.3 Tools

The following suite of tools were selected for inclusion into the course.

MICON (Board synthesis).

MICON is a tool suite for design of digital electronic systems. M1 is a knowledge-based synthesis engine for generating boards. CGEN is a knowledge acquisition tool that allows hardware designers to increase the knowledge base of M1. From high-level specifications, a set of constraints and a knowledge base, M1 can generate a complete netlist. In the past few years, MICON was used to generate computer workstation sized board designs. The MICON tool suite also includes a commercial schematic editor. [Birmingham et al, 1989] [Gupta et al, 1990]

ABLOOS (Placement).

ABLOOS is a framework for 2-D layout design. ABLOOS uses a graph representation of spatial relationships between rectangles and generation rules to produce alternative arrangements. It then uses a knowledge base evaluation module and a branch and bound search technique to find layouts minimizing the number of constraint violations. Unlike optimization tools, ABLOOS provides users with more than one solution, allowing them to visualize possible trade-offs. ABLOOS also allows hierarchical decompositions of layout problems along with various solution policies (e.g. top down, bottom up, etc.). ABLOOS is used in various domains, such as the layout of analog circuits boards and service cores in high-rise office buildings. [Coyne, 1991]

NOODLES (Solid modeling).

NOODLES is a non-homogenous, non-manifold solid modeler. It allows the representation of one, two and three dimensional objects. It is currently used as a representation for various tools such as an injection molding critique, an assembly analyzer and an assembly robot arm trajectory planer. The NOODLES calculator allows users to describe solid models in a CSG (constructive solid grammar) representation, translate them to a boundary representation, and visualize the results. [Gursoz, 1991] [Gursoz et al, 1991]

GENESIS (Solid generation).

GENESIS is a tool for generating classes of solids from a boundary solid grammar. A boundary solid grammar is a set of rules whose left hand side is a predicate describing a property of a solid model and whose right hand side is an operation on the boundary of the solid. Boundary solid grammars have been written to describe Queen Ann houses, support structures for injection molding, etc. [Heisserman, 1991]

Other tools would be used in demonstration mode because of the unavailability of multiple copies of their computing platform, or their relative immaturity and lack of robustness[1]. The tools to be demonstrated included a thermal analyzer, an assembly analyzer, and an injection molding critique.

1. In this document, we define robustness as the inverse of the average number of failures experienced by a novice user while using the tool. By failures, we mean every failure experienced by the user during the operation of the tool, whether it was caused by the tool, the system or the user itself. See Section 4.5.1 for a complete definition.

2.5 Project team organization

The preparation and teaching of the project involved one senior faculty member, a secretary, three full time TA (teaching assistants), an industrial designer, three undergraduate REUs (Research Experience for Undergraduates) and up to six other graduate students at various phases of the project.

The three TAs also functioned during the design process as an electronic designer, a mechanical designer and a software designer. The electronic designer was responsible for coordination during the design process. Each of the TAs and the industrial designer were responsible for presenting in the lectures the portion of the artifact they designed. The software designer was also responsible for the tool integration effort and the coordination between tool providers. The REUs were involved at various points in the manufacture process and the development of the software of the artifact. The faculty member was responsible for the project lectures and the coordination with the rest of the course.

The on campus personpower needed for the whole project added up to 468 person days (pd) (1 person day ~ 10 hours), split as depicted by Figure 6.

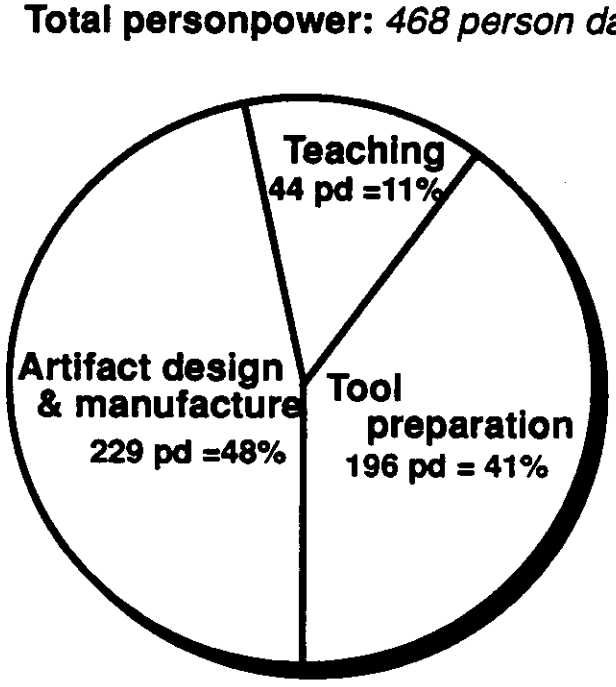


Figure 6 Personpower per task

3 Design Process

3.1 Overview

This section describes and analyzes the project team design process. This section is structured as follows. First a brief description of the project design team and the final version of the artifact are given. The resources and personpower required to complete the artifact are described next. Then the chronology of the design process and the interactions between the designers is discussed. Finally, the flaws encountered during the design process, a brief discussion of their cause and alternate solutions are presented.

The purpose of this section is twofold. First, to provide enough data to allow an estimation of the time and resource needed to design an artifact of similar size. Second, to identify common problems encountered during a design process.

The data presented in this section has been collected from various sources, such as e-mail messages exchanged between designers, field notes, time stamps of data files and informal interviews of the designers.

3.2 Project design team

The project design team was composed of seven students and one faculty member (see Figure 7). The electronic designer was responsible for the coordination of the design process. The faculty member made cost sensitive decisions from a set of proposals provided by the electronic designer. The mechanical designer and the industrial designer worked together on the design of the housing. The three REUs were involved at various points in the manufacture process and during software development.

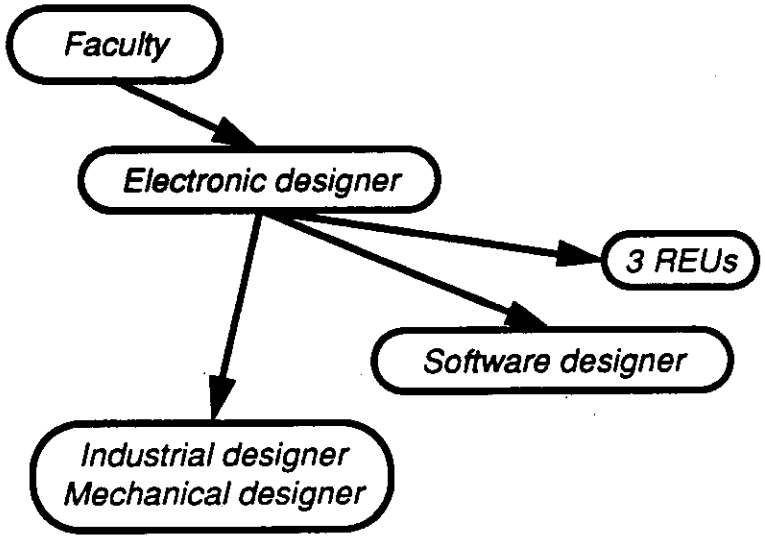


Figure 7 Project design team organization

3.3 Artifact

The final version of the artifact, depicted in Figure 8, had the following characteristics:

- CPU board:
  - 80188 CPU
  - AT bus
  - 8kBytes RAM
  - up to 8 x 64kBytes EPROMs
  - 12 chips of random logic
- Private Eye CGA adaptor board (purchased with the Private Eye)
- 2 1/2 x 5 1/2 x 12 inch vacuum molded enclosure
- Powered by 8 x 1.5 V batteries or a 12 V DC source
- Three application push buttons, one on/off power switch, and one reset push button

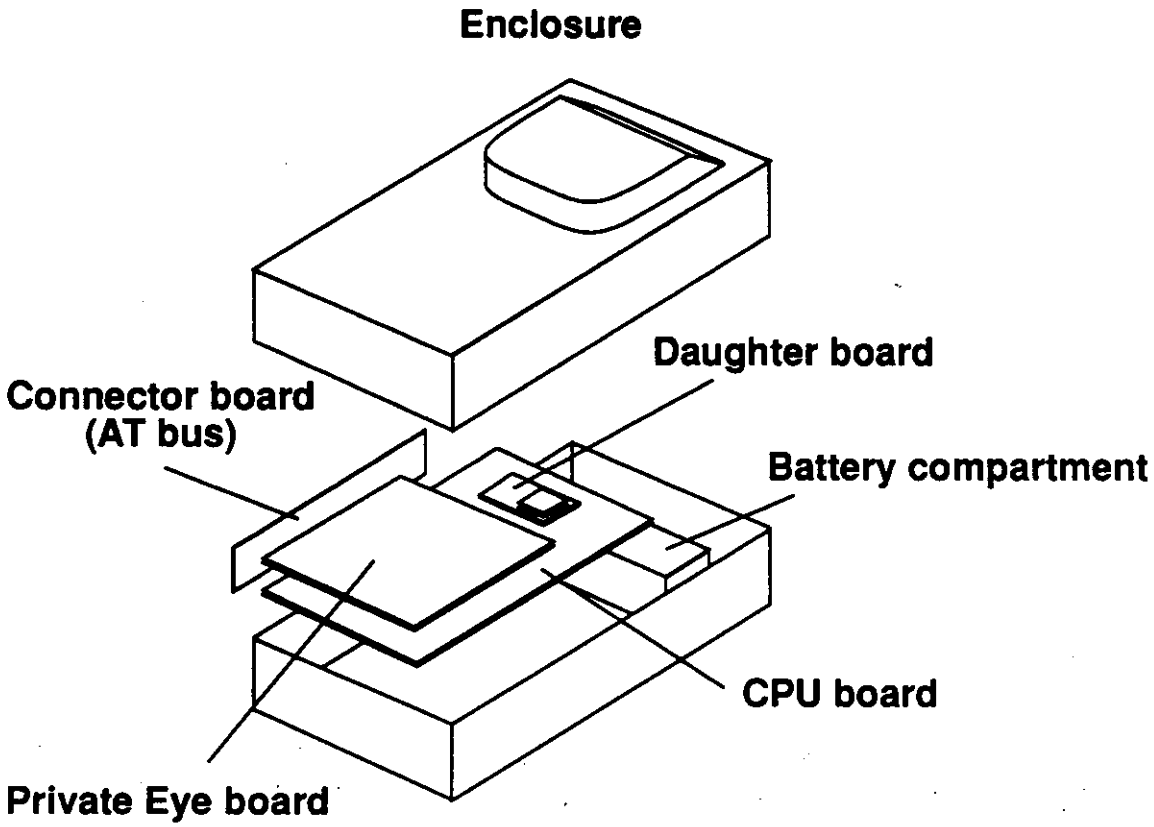


Figure 8 Disassembled view of the artifact

3.4 Personpower

The complete design and manufacture process took 229 person days. This number includes the early prototypes whose development was interrupted (such as the first prototype housing and the first version of the software), but does not include the time spent to present the various design phases in the project lectures. In other words, this number can be interpreted as the personpower required to complete the design and manufacture of an artifact of this scale, independently from the fact that this design was completed in the context of a course.

Figure 9 shows the various tasks and the personpower involved in the design and manufacture of the artifact. The vertical axis represents calendar time. Each bubble is a task. The width of the bubble is the maximum number of persons who worked on the same task. Bubbles next to each other represent concurrent tasks. The area of the bubbles does not directly correspond to their personpower, since there was not usually someone working full time on a single task from start to completion. The gray rectangles below the bubbles corresponds to the two six week segments of the course.

Figure 10 shows the main tasks of the design process with respect to the designers responsible for them. The vertical axis represents time, but unlike in Figure 9, the height of the bubble has no meaning. Each bubble represents a task. An arrow between two tasks represents both a precondition in time and a communication point between designers (or manufacturers). An arrow interrupted by a lightening symbol represents a communication point where information was misunderstood or incomplete. The thick gray arrow lines represent the critical path of the design process.

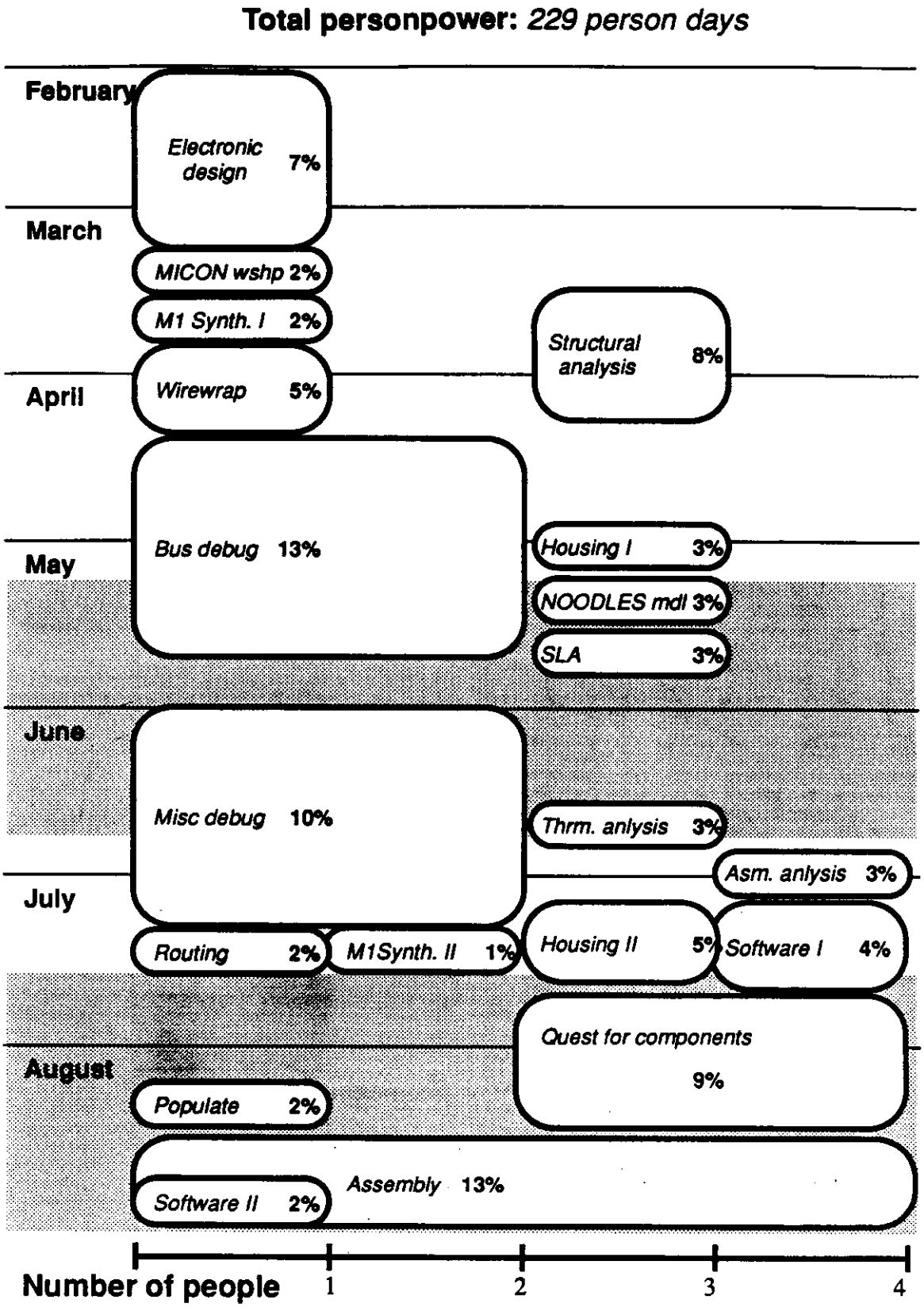


Figure 9 Design process and personpower

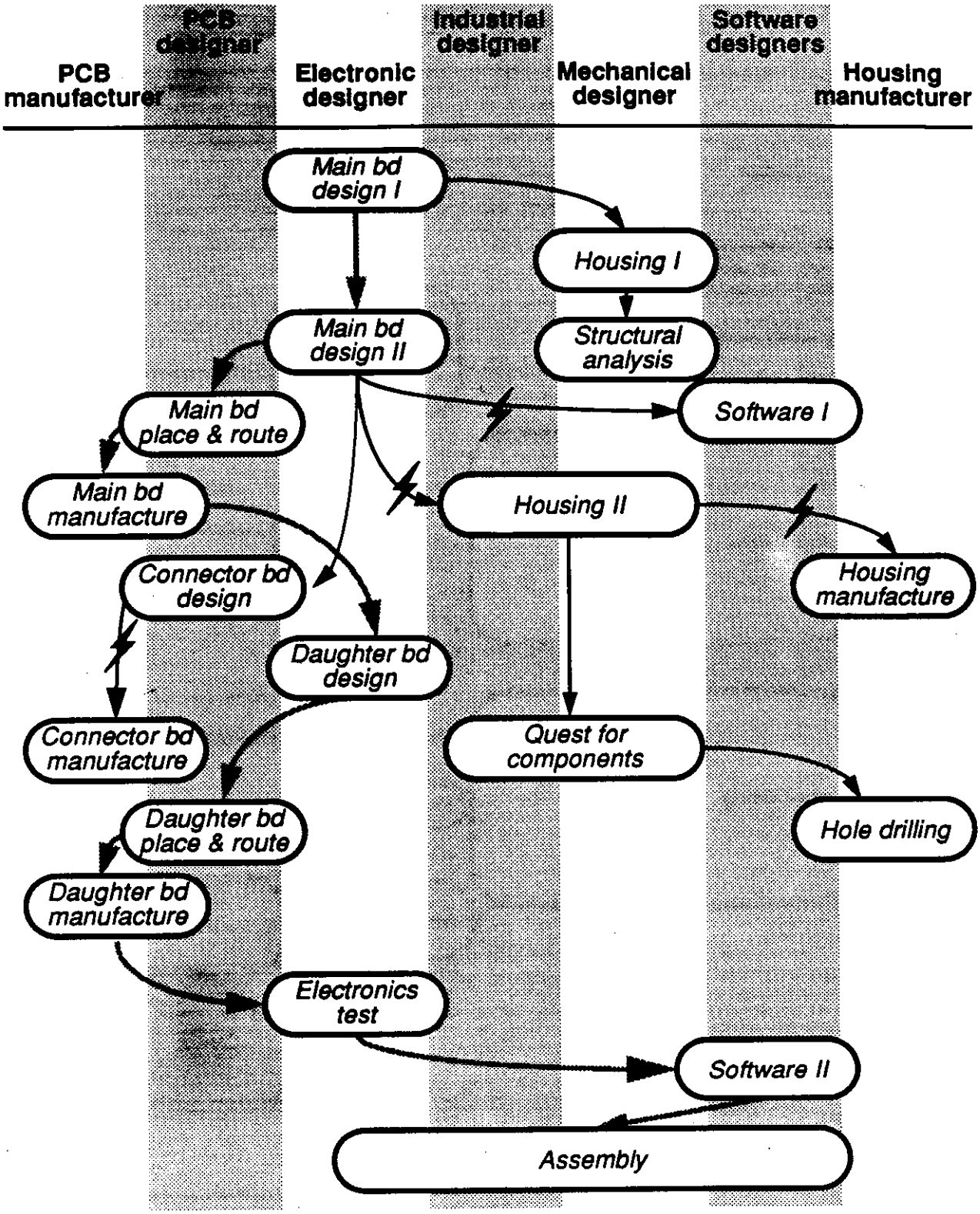


Figure 10 Design task graph and communication errors

Electronic design, MICON workshop and M1 synthesis

The design process started with the hand design of the electronic portion of the artifact. An initial hand design was used to estimate the materials cost for the project. This design was improved after the critique of off campus personnel who had previous design experience with the AT bus.

The resultant design served as the basis for training cases on parts not already in the MICON knowledge base. In March, the designer attended a workshop given to an industrial affiliate, and then synthesized a second design using MICON.

Wirewrap and first prototype

A first prototype was realized as a wirewrap board. The wirewrap took two weeks, followed by a one month debugging period before the board was able to draw lines on the display. The main problem encountered during this first debugging period was incorrect bus timing. Accurate timing information of the Private Eye adaptor board was initially not available, and was later provided by Reflection Technology.

It took one more calendar month before the first prototype was declared working. The personpower available for the design during this period decreased, because the main designer was involved in the course.

The knowledge base of M1 was updated in July to reflect the changes made during the debugging period and a second synthesis of the board was performed in two days.

Placement, routing and second prototype

The chip placement was done by hand and the routing by a commercial tool. The layout task lasted four days. The person who performed this task was an expert in placement and routing, which explains the short time needed to complete the task. The CAD/CAM files were then sent to a printed circuit board manufacturer. Three boards were manufactured in four days, the remaining twenty in two weeks.

A second prototype was built with the printed circuit board in order to uncover any errors introduced during the layout phase. One major error was discovered: the pinout of the processor was incorrect. This was due to a human error while entering the pinout of the processor part in the database. Since the printed circuit board could not be manufactured again, a small board (called the daughter board, see Figure 8) about twice the size of the processor was then designed and manufactured off campus. The daughter board was composed of two chip sockets, one for the processor, and the other for the original chip holder in the board.

A third board (called the connector board, see Figure 8) was ordered and manufactured off campus. The connector board formed a short AT bus between the main CPU board and the adaptor card. After the order was shipped, one more error was discovered. Two bus lines unused by the main board were not connected. A figure describing which lines to ground was faxed to the manufacturer. Unfortunately, the manufacturer incorrectly interpreted the figure as a mirror image. The connector boards were repaired on campus.

Population and soldering

The boards were populated in one day by three REUs who worked at EDRC during the summer. The boards where then sent off campus for wave soldering. The daughter boards were also sent off campus for wave soldering. The daughter board, unlike the main board did not have a soldering mask. The manufacturer wave soldered one side, and hand soldered the other side, resulting in a longer manufacturing time. Out of thirty daughter boards, only eleven were working after the wave soldering. The remaining boards had solder bridges which shorted processor signals. The non-functioning boards were repaired on campus by hand.

Housing, NOODLES model and SLA

An initial requirement of the artifact was to be able to fit in a radio/cassette car rack. The initial design of the enclosure included an off the shelf metal cage and a front piece manufactured by the SLA. Shortly after the beginning of the course, it was decided that the artifact would not be a display for road maps, but for blueprints instead (see Section 2.4.2). The requirements of the enclosure changed and the requirements that the product is carried over the shoulder was added to the artifact specification.

A second design was done, based on a plastic housing purchased off campus which would be completed with a front piece including a three dimensional EDRC logo. A NOODLES model of the front panel was built and sent to the SLA for manufacturing. After one week, it was realized that the SLA was not accurate enough along the vertical axis for the EDRC logo. The manufacture of thirty copies would also have posed a problem with respect to time.

A third design was constructed from styrofoam by the participants of the course, and sent off campus for manufacturing by vacuum molding manufacture. Two full time people then spent 10 days looking for off the shelf components such as switches, push buttons and shoulder straps. Following that period the location and dimension of the holes were sent to the manufacturer of the housing.

Software

The first version of the software was developed in C on a workstation, before the main CPU board was fully functional. An REU scanned a blueprint, translated it into a reasonable format and wrote scaling and scrolling software functions.

The board was fully debugged only shortly before the REU left EDRC. The software port to the board was done by a second person. Two days were spent porting the software. Following this period, it was decided to start the software over, because of the slow performance of the first version. The development of the second version required four days. Because of the lack of time, the functionality of the software was reduced. Only one bitmap was stored in the artifact, and the "zoom in" function was suppressed (i.e. it was not possible to display the blueprint in a larger scale than the scanned bitmap).

Assembly

Twenty five artifacts were manufactured. Up to seven persons worked for one week on the assembly process. By the end of the course twelve artifacts were fully functional.

The low yield was primarily due to the lack of quality control during the manufacturing process. Fragile parts such as the daughter boards and the processors were the only ones tested before assembly. Later, it was realized that the main board should have been tested at every step of the assembly. For example, about a third of the assembled artifacts had a non-functioning RAM or a non-functioning processor. Both components were probably damaged by static electricity.

Most of the mechanical errors were discovered during assembly. The most annoying (and time consuming) flaws were the omission of the reset button, the battery compartment lid design flaw, and the size of the heat sink.

The reset button is one piece of information which slipped between the cracks when the electronic designer passed the digital design to the industrial designer. The electronic designer probably did not mention the existence of the reset button since it was something obvious in the electronic domain. The industrial designer did not think about it because of its non-obvious function in the mechanical domain. The result of this misunderstanding was that the design of the enclosure did not include this button. An emergency order was given to the mechanical shop of the ECE department to drill an additional hole in the enclosures. Because of the rush associated with this patching, the location of the new hole was inappropriate: the pins of the reset button were touching the program EPROM and threatened the correct functioning of the artifact (see Figure 11). The pins of the reset button were bent, and a piece of tape was attached to the EPROM to avoid short circuits. After the pin bending was done, the wires connecting the switches had to be resoldered.

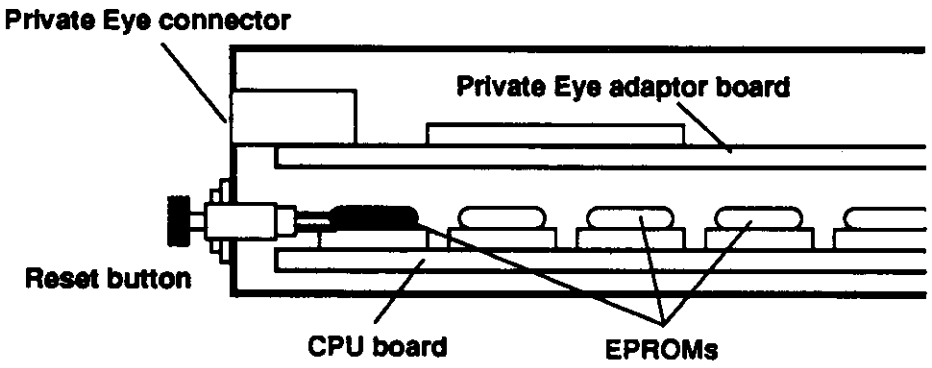


Figure 11 Reset button misplacement

The initial design of the battery compartment planned to glue the battery holder to the lid, and attach two stripes of velcro on the bottom of the compartment and the battery holder. A glue could not be found which would bind the lid to the battery holder with more strength than the battery holder was bound to the battery compartment. A quick solution to this problem was to wrap a logo sticker around the lid in order to slightly increase its width. The problem with this solution is that the lid is difficult to remove without tools (see Figure 12).

The heat sink component is located at the end of the CPU board near the battery compartment. The component specified during layout and mechanical design was not available during the manufacturing process. Instead, a larger component was order. During the assembly, it was realized that the wings of the new component prevented its insertion on the board. The wings were bent (see also Figure 12).

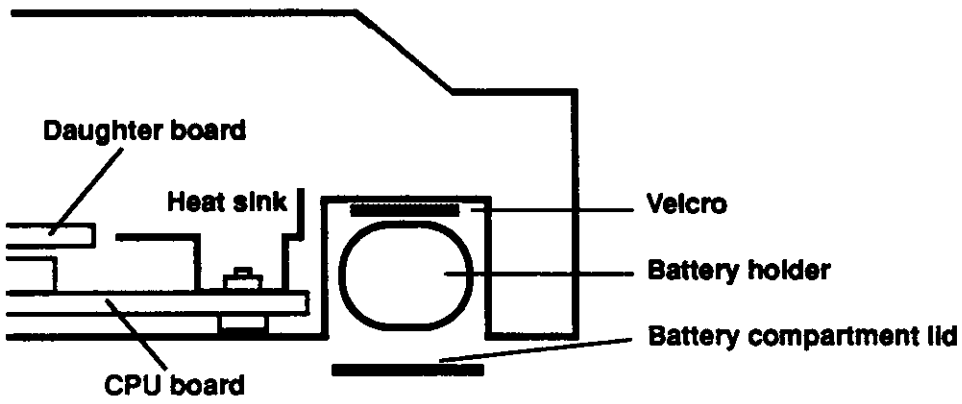


Figure 12 Battery compartment and heat sink

3.5 Flaws and unforeseen problems

Electrical flaws	Cause	Immediate solution	Time lost [days]
Processor pinout	Incorrect entry in database	Daughter board	7
Low yield of the daughter boards	Lack of soldering mask	Manual cutting of solder bridges	1
Connector board bug	Wrong interpretation of spec. by the manufacturer	Manual addition of components to connector bd.	1
Capacitors polarities reversed	Human error during board layout	Swapped capacitors	0.5
Pinout of the switches	Human error during assembly	Resoldered switches	2

Mechanical flaws	Cause	Immediate solution	Time lost [days]
Battery compartment lid	Design error	None	
Reset button omission	Misunderstanding between designers	Drilled additional hole on campus	0.5
Reset button pins touching pgm EEPROM	Cascaded from previous problem	Taped EEPROM	0.2
On/off labels (or LED) on power switch missing	Design error	None	
Heat sink too wide	Unavailable component	Bent heat sink wings	0.2

Software flaws	Cause	Immediate solution	Time lost [days]
Poor portability between workstation and PC	Lack of software experience	Second version of software	12
Poor portability between PC and CPU board	Compiler bug	Hack around	1
General flaws			
Quality control	Lack experience in manufacturing		10
Spares	Underestimation of the number of defective components		15
Total time lost [days]:			**50.4**

Table 1 Problems encountered during the design process

3.6 Manufacture outside campus

Item ordered	Out	In	Time [days]
Private Eyes	4/1	4/12	11
21 80188 processors	4/3	4/5	2
9 80188 processors	4/3	4/8	5
Parts (w/o processor)	7/2	7/12	10
Capacitors/resistors	7/12	7/17	5
Spares	8/27	9/6	10
3 main boards	7/9	7/13	4
27 main boards	7/9	7/17	8
6 main boards	7/9	7/24	15
Daughter boards	7/29	8/5	7
Connector boards	7/19	7/26	7
Wave soldering	7/27	7/30	3
Housing	7/29	8/19	21

Table 2 Order times

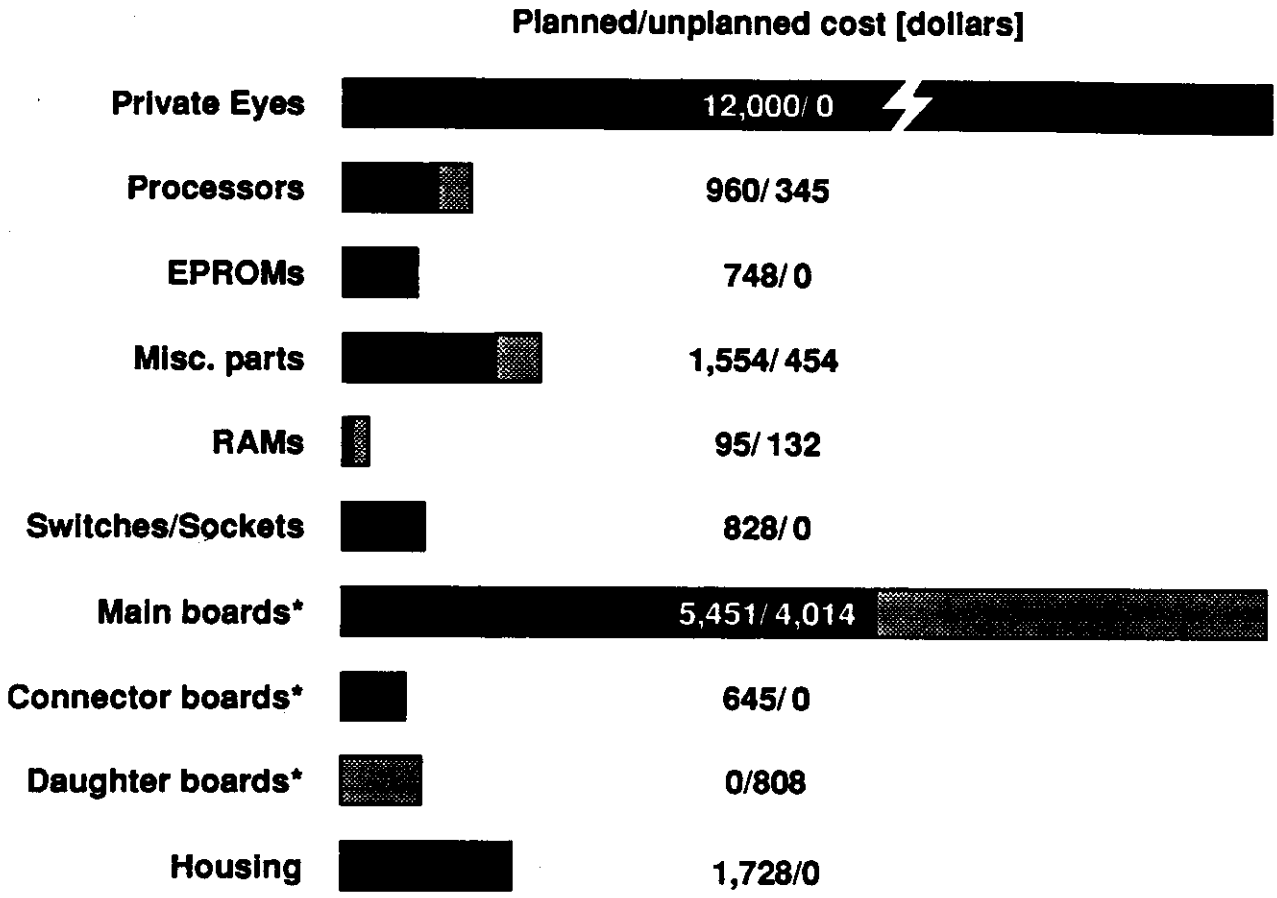
Table 2 describes the delivery times for the parts and work performed off campus. Dates do not include the lead time needed to obtain a purchase order. In the case of the Bosch course 91, the lead time was shorter than average (two days; average: one week). The high variance on the delivery of digital components (two days for the first batch of processors, ten days for the spare parts) was due to the unavailability of parts in stock.

The main board was manufactured in three batches. Three boards were ordered with a three day delivery and were used for testing purposes. By the time the testing was completed, twenty seven more boards were received. Six additional spare boards were manufactured with a regular three week delivery. Initially, the main boards were to be manufactured in a single three week delivery batch, which would have been cheaper (see Figure 13). The delays in the design process and the approaching end of the course dictated the accelerated delivery strategy.

The housing was vacuum molded off campus. The manufacturer was provided with drawings of the housing without the specification of the holes. At the time the specification was handed to the manufacturer, the industrial designer and the mechanical designer were still looking for switches and shoulder straps (see Figure 9). The hole specification was sent later, which delayed the delivery of the housings for one week.

Figure 13 represents the cost of the material and off campus personpower spent for the manufacture of the artifact. The black bars represent the costs as they were planned before May 91; the gray bars represent cost incurred by contingencies or design flaws. Twenty nine artifacts were eventually manufactured.

The additional costs for the processors, RAMs and miscellaneous parts were caused by an unexpected number of damaged parts during the assembly process. Ten processors and seven RAMS were damaged by static electricity.



Total cost: *$24,009 + $5,753 = $29,762*

Cost per artifact: *$29,762 / 29 = $1,027*

Figure 13 Cost and contingencies

3.7 Towards a better design process

Coordination

An unusual aspect of the design process was that no design meeting involving all the designers ever took place. The first time the design team was assembled in its entirety to discuss the artifact was at the beginning of the manufacture process. Most of the coordination of the project was done through bilateral communication between the designers (often by e-mail), which induced a smaller overhead.

A drawback of this policy was that the redundancy in the communication was small, allowing some important information to slip between the cracks. A third peer present during these communication exchange could have allowed additional cross-checking and recording of the information transmitted. Flaws such as the omission of the reset button were probably due to this lack of verification. Another drawback of the peer to peer communication was that the time to update other members of the team about less important issues was long, which slowed the design process when any of the designers was absent (e.g. conference trip, job quest, etc.).

An immediate solution for improving consistency, completeness and recording of the information exchanged between designers would be to restrict the medium of communication to e-mail, and, for a small design team, to broadcast messages to all members of the team. If the mail traffic becomes too high, the designers might not read all the messages as thoroughly as they should, thus reducing the benefits of the redundancy. This problem could then be solved by having the chief designer (or any member of the team who has a multidisciplinary background) responsible for centralizing all the messages and forwarding them to the persons of interests. The chief designer could also create a taxonomy of messages by importance and domains in order to mechanize the forwarding process.

A potential second solution is to make available to the designers the structure of the design information space. An example of this approach adopted by the n-Dim project [Subrahmanian et al, 1989] could serve as a test bed for sharing and structuring of the design information. In this scenario, the designers would be able to access the information they need and would also be notified of changes in the information that they choose to monitor.

Any proposed solutions would have to be tested and refined before substantiating any claims of reduction in overhead to traditional methods. Such issues as how easily designers would adapt themselves to new communication schemes are not easily predictable.

Consistency

Many flaws were the result of an incorrect change of the design after a previous flaw. The pins of the reset button were bent and taped due to a misplacement of the reset button; the heat sink wings were bent due to the change in choice of a component. Generally, late changes of a design are more likely to introduce errors because they are often made outside the original context of the design. Many assumptions made by the designer are no longer available at the time of the change. In the case of the heat sink, the electronic designer did not know that the heat sink was located adjacent to the battery compartment, and thus failed to update the mechanical designer about the change.

Both solutions proposed in the previous subsection (e-mail and nDim) would provide the designers with a history of their decisions and assumptions. However, nDim would allow a more efficient retrieval, since nDim allows structuring of the design information. In either case, the designers will be able to maintain the consistency of the design only if they state explicitly as many assumptions as possible. This task is not trivial, since the designers do not know in advance which constraints are likely to be violated in the future.

Confidence in tools and databases

Most of the CAD tools used during a design process are complex and hard-to-use pieces of software. The consequence is a non zero probability of introducing errors in the design information base either because of software bugs or operational errors. This probability is even higher for research software such as that used during our design process. For example, the most costly design flaw was the processor pinout error.

These types of flaws are hard to detect before the manufacture of a prototype. For a design process which would involve more than one person of a domain, a solution would be to introduce a cross verification mechanism. In the example of the processor pinout flaw, the person who would do a sanity check before sending a netlist to the routing could be a designer different from the one who entered the pinout information in the database.

Another class of time consuming faults was caused by the confidence in the tools we used. A simple compiler bug delayed the software development for two full days, mainly because the software developer assumed the compiler was bug free.

Prototypes

Some flaws were time consuming not because of their seriousness, but because they were discovered late in the manufacturing process. For example in order to overcome the heat sink flaw, we only needed to bend the wings of the component. However, we had to bend thirty heat sinks. A similar case was the floating lines of the connector board.

A possible solution to detect more design flaws before manufacture would be to build more prototypes. We limited the number of prototypes because of the lack of time, but in hindsight, we believe we wasted almost as much time correcting late design flaws than we saved.

From our previous experiences in design, the danger of a prototype based design process is the temptation of degrading to a 'trial-and-error' methodology. Introducing more prototypes increases the level of confidence of the designer in the early discovery of design flaws, thus lowering his use of other fault prevention mechanism (e.g. peer verification). Protoypes should be viewed as an additional test rather than a replacement. Another drawback of a prototype based design process is a longer calendar time.

4 Tool Preparation

4.1 Overview

This section describes and analyzes the preparation of the tools for the course. A description of the tasks and personpower performed during the tool preparation is given first. Next, the problems encountered are presented along with a brief discussion of their cause. The remainder of the section proposes a few short term and long term problem solutions.

The purpose of this section is multiple. First, to provide enough data to allow a more accurate estimation of the personpower and resources involved in the preparation and development of CAD tools for a similar course. Second, to identify common problems encountered during a software integration process. Although research software was being integrated for a course, we claim that similar problems can be encountered in an integration effort which would take place in industry. The third purpose of this section is to identify problems and solutions in the development process of the tools themselves.

The data presented in this section was collected from documents written during the integration evaluation, e-mail messages, field notes and source code file time stamps.

4.2 Tool preparation team

The tool preparation team was composed of two graduate students and a faculty member (see Figure 14). At various points of the tool preparation, the developers of the tools were also involved. The software designer was responsible for the tools and the integration effort, while the tool providers were responsible for bug fixes and the implementation of new features. The mechanical designer participated in the integration evaluation. The EDRC faculty made the choice of tools and architecture from a set of proposals provided by the software designer.

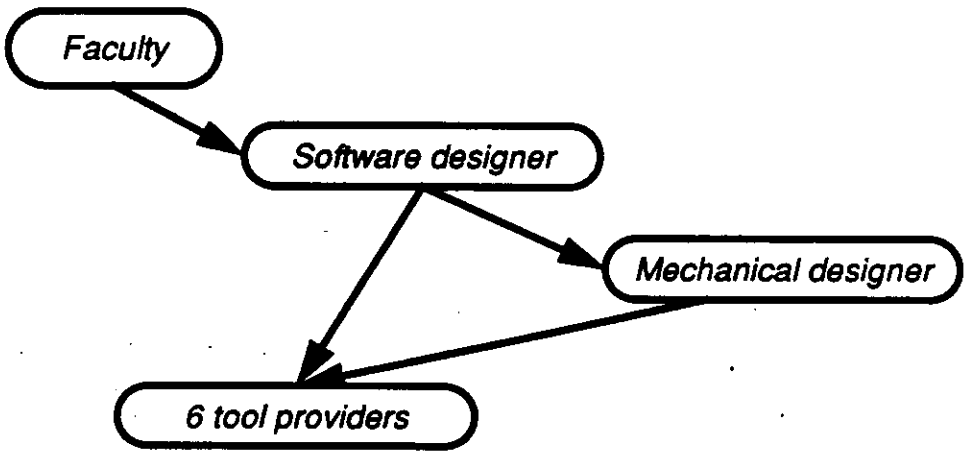


Figure 14 Tool preparation team organization

4.3 Task organization and personpower

Figure 15 shows the personpower spent for preparing the tools for the course. The vertical axis represents calendar time. The gray area in the background corresponds to the two six week segments of the course. Each bubble is a task. The width of the bubble is the maximum number of persons who worked on the same task. Bubbles next to each other represent concurrent tasks. The area of the bubbles does not directly correspond to their personpower, since there was not usually someone working full time on a single task until its completion. The task names with a superscript star corresponds to tasks which were interrupted before completion.

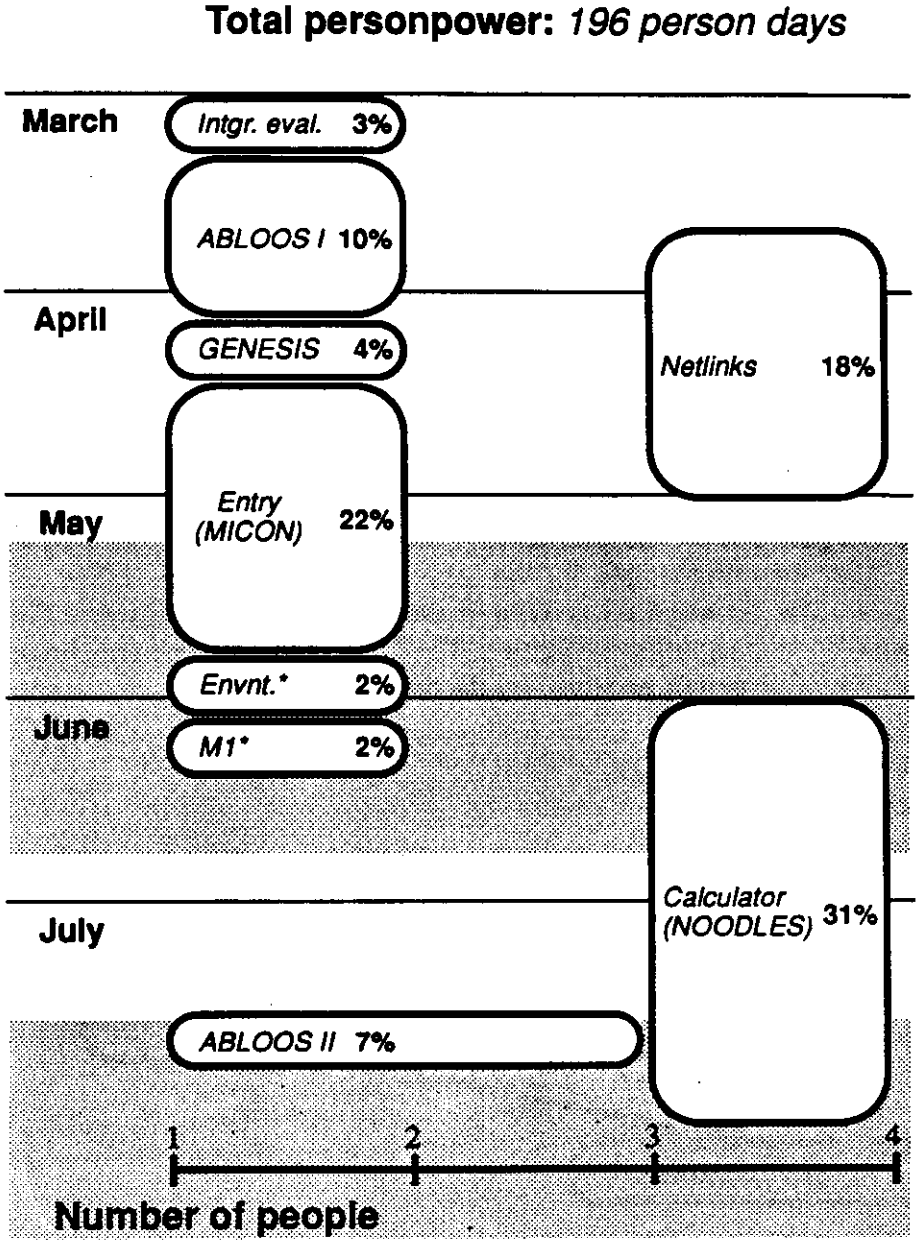


Figure 15 Personpower spent on tool preparation

The tool preparation took 196 person days. This number includes the time spent on tools which were eventually not used in the course. It includes the time needed for the development, test and port. The following paragraphs discuss each bubble individually.

Integration evaluation

The initial target was to build a single environment in which the participants would design the artifact presented in the previous section. The first step of the tool preparation was to collect the following information about each of the candidate tools:

- tasks performed
- architecture requirement
- operating system requirement
- display requirement
- input/output formats

The architecture/operating system/display requirements were needed to select a platform to use for the course. We also wanted to know the input and output formats of the tools to estimate the number of translators that needed to be written.

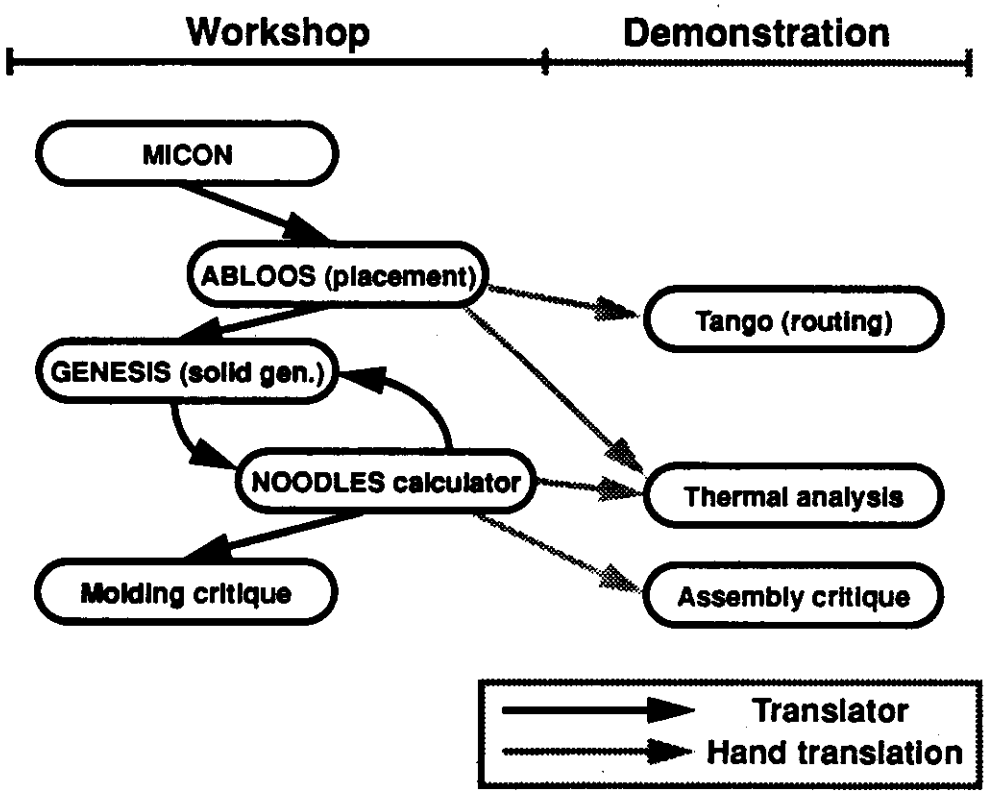


Figure 16 Target environment

The evaluation was solely based on the information collected from the tool providers. We did not have time to actually be trained on the tools and use them before the decision was made. At the end of the evaluation period, we decided to realize the environment depicted by Figure 16.

MICON would be used to synthesize a digital design. Geometric information about the parts and their connectivity would then be sent to ABLOOS for placement. Once the dimensions of the board were known, an enclosure would be generated with GENESIS. The participants would also be able to hand build an enclosure with the NOODLES calculator. The final NOODLES model would then be sent to the injection molding critique, the assembly critique and to thermal analysis. The tools in the right column were tools to be demonstrated only.

We decided to use as a platform a DEC 3100 running AndrewOS with a monochrome display. Performance critical tools such as ABLOOS would be run remotely on a higher performance workstation (e.g. a DEC5000 with 40MBytes of memory). Four color monitors were purchased for the tools requiring color.

The environment needed the following software work to be completed:

- M1, update knowledge base
- ABLOOS, develop a constraint base
- GENESIS, port from HP to AndrewOS (wireframe display) and develop a grammar
- NOODLES, finish development of the calculator
- GENESIS to NOODLES translator
- MICON to ABLOOS translator
- NETLINKS, cleanup

At this point in development, we did not know the computer expertise of the participants. Some of the tools we investigated had a complex interface. They required the user to write constraints or rules in a programming language (e.g. GENESIS: Prolog; ABLOOS: Lisp), or they required the user to do "file reshuffling" (i.e. explicitly invoking translators, moving or renaming files, etc.). We decided to built an environment displaying files and tools graphically, in which translators invocation and file renaming would happen transparently. We also planned to provide a graphical user interface for the tools which required it the most. We started to work on the tools which presented the most uncertainties.

ABLOOS I and II

The ABLOOS I block represents the time needed to learn about its mechanics and to built a reduced placement example based on the hand design of the board. The ABLOOS II bubble is the time needed to upgrade the first example to the final design of the board. During this phase the tool provider also rewrote the knowledge base we developed in a more robust and generic fashion. The CPU time needed to find a good solution was also reduced.

GENESIS

Although GENESIS was one of the most robust tool of the environment, it needed significant work for integration. The original version of GENESIS uses specific 3D graphics hardware and runs under HP/UX. The software needed to be ported to the platform used in the course. The user interface and a wireframe rendering module were upgraded. The bubble also include the time needed for the tool provider to write a translator to the NOODLES file format.

Entry (MICON) and M1

The MICON tool suite did not need to be ported. On the other hand, the MICON tool suite does not provide much support for correcting the knowledge base of M1. Since we expected an iterative design cycle in the framework of the course (the participants did not have much experience with either digital design or the MICON tool suite), we decided to write a graphical browser to access the database components. This tool allows an easier visualization and correction of the information stored in the database. Although the time needed to develop this tool was shorter than usual, it was still significant. The browsing functionality of the tool was ready for the MICON workshop, but the tool was fully functional only two weeks later.

Part of the user interface of the browsing tool was to be reused for M1. The development of the user interface of M1 was interrupted after it was realized that it could not be completed on time.

NOODLES calculator

The NOODLES calculator is a graphical interface allowing the user to specify a solid in terms of a CSG tree and visualize it. When the tool evaluation period started, a prototype of the NOODLES calculator built on top of Motif[1] was already available. The tool presented many robustness problems, and later, its developer stopped working at EDRC. It was then decided to rewrite a second version on top of a more robust user interface library. Its development was completed only after it was decided to reduce the number of workshops in the course (see Section 5.2).

NETLINKS and Environment

The target environment planned a mini framework including functionality to remotely invoke tools, manage data files and translators. NETLINKS is a library providing functionality for remote process and file management which needed to be cleaned up before it could be used in the course. NETLINKS was completed on schedule.

The development of the graphical interface to the environment and the control portion of the mini framework was interrupted after it was realized it would not be completed on time. Only the GENESIS to NOODLES translator was realized.

1. Motif is a toolkit from the Open System Foundation for building graphical user interfaces.

4.4 Flaws and unforeseen problems

The password file on the Andrew workstation is stored on the local disk. This file is updated whenever the workstation is rebooted (which happens often). The first day of class, some of the Andrew workstation had not been rebooted since the creation of the participants accounts, who were unable to login on some of the workstations. The problem was identified and solved about an hour after the class started.

The performance requirements of the tools were not taken into account during the integration evaluation. In the case of the MICON database, this lead to serious robustness and performance problems. The database server usually runs on an Ultrix/DEC3100. The database was duplicated four times and each database server was assigned a different Andrew workstation. The synthesis process was about two to three times slower than usual. The next lecture, we run the four database servers on the original DEC3100 and increased performance. The workstation crashed after twenty four hours. Doubting that the database servers were the main reason of the crash, we rebooted the machine and restarted the database servers. After another twenty four hours, a user of this machine asked the facilities to kill the database server processes, because they were occupying a large disk swap space and seemed abandoned. Neither the user nor the teaching assistants of the course were aware of the others' use of the workstation.

The commercial schematic editor used by MICON had a license for 30 sites. A license server was running on a separate machine. It would give out a token to any starting copy of the editor, and get the token back when the editor was exited. On the first day of the MICON workshop, the participants exited the editor by killing the process from the shell. In such cases the token are only released after a certain time-out (set by default to twenty four hours). Before the end of the class, no more license token were available, and none of the participants were able to start a new copy. After reading the administrator's manual of the schematic editor, we decreased the time-out to five minutes.

Some of the tools we used (such as the OPS83 compiler for MICON) use the /tmp directory on the local disk as a temporary storage. On Andrew workstations, there is no /tmp directory, and the users account is used instead. The quota of the participants' accounts was limited to 1 MByte, which was insufficient for most of the tools. We instead used an Andrew file server partition instead as temporary storage, but forgot to modify the access privilege to allow the participants to access it.

The initial estimation of the personpower needed for preparing the tools for the course was well underestimated (famous last words). Most of the work was done by students who were initially not related to the tools used. For example, the tool providers of GENESIS and ABLOOS were defending or finishing their thesis, thus they could not spend a long time in the preparation of those tools. Another reason is that the robustness of all the tools was overestimated, partly because we did not spend the time during the evaluation process to become familiar with and use the tools ourselves. Lastly the people (among which, the authors) who did the personpower estimation tend to make optimistic estimations and did not take into account contingencies. See Section 3 for a discussion of this problem.

4.5 Towards better tools

4.5.1 Short term solutions

The major problem in the tool preparation was that we made several decisions (e.g. choice of tools, choice of architecture) with incorrect and insufficient information. This led to incorrect estimation of person power requirements and in software that was not robust enough for teaching. The next paragraphs focus on which information would have been useful to those decisions, and how to collect it.

Integration evaluation

Even if the tools used in a course project are not to be integrated into a software backplane, an integration evaluation is still necessary. It is likely that a project will make use of only one kind of workstation, first because the availability of resources, second because there is not enough time for participants to become familiar with more than one environment.

Information to be collected are:

- Operating systems/architectures the tool can run on
- Display required (color, monochrome, which version and release of the window manager, etc.)
- Load requirements (memory and CPU time)
- Disk space requirements

We did not collect precise information about the last two items. Collecting those would have prevented the MICON database crash (see Section 4.4), which was due to a full disk partition.

All the collected information has to be verified carefully. For example, knowing that a tool should run under Ultrix 4.0, because it was developed under Ultrix 3.1 and Ultrix is upward compatible, is not sufficient. There exists incompatibilities between successive releases of any software or architecture, and there are often hidden bugs which may be triggered in a different environment.

The load and disk space requirements have to be evaluated in the context of a class. For one tool suite, having twenty six people using a tool may mean twenty six copies of a tool accessing the same database. Extrapolating the performance requirements of twenty six copies of a tool from a single copy is misleading. Even though the use of the tool is distributed, the database becomes the bottleneck. This situation may actually degrade the performance with respect to a single user to the point of a system crash.

In the event of integration into a software environment, the following additional information is also needed:

- Tool invocation sequence
- Data file formats

The invocation sequence will determine whether the tool invocation can be automated or not. In some tools, specifying an input file can only be done interactively (i.e. by the user after the invocation of the tool). Knowing the data file formats determines how many translators are missing. One should keep in mind that developing a correct translator may take as long as the development of a tool. The test of a translator is long, because of the large number of special cases, and the frequent unavailability of a precise specification of the formats involved.

Robustness evaluation

After the Bosch course 91, the project team agreed that the robustness of tools is an important factor in efficient teaching. The equivalent of a few classes were lost because of system crashes, without mentioning the frustration experienced by the participants. Most of the participants of the course had little or no recent experience of CAD tools, which caused (from our point of view) them to have an ideal view of the robustness of tools in general.

The concept of robustness and its measure has yet to be defined. This paragraph proposes a few metrics for measuring robustness in the early preparation of the course. A common metric for evaluating the robustness of a system is the measured mean time to failure (MTTF). However, the MTTF of software is highly dependent on its use, and the expertise of the user. One can expect that the developer of a tool will experience a longer MTTF during operational use since they have a good insight into which portions of the software are fragile. Novice users often show behaviors unexpected by the tool developer (e.g. during the REU training on the MICON database browser, one REU repeatedly double clicked on some areas of the window, expecting a behavior similar to some Macintosh applications; this action caused the tool to crash; this sequence of mouse clicks was never tested by the developer, since this behavior did not correspond to a tool command).

Another issue related to robustness is the behavior of the system in case of user errors. Although the system may not crash in case of an incorrect behavior of the user, it may sometimes either not report the error or report it incorrectly, leading to unexpected responses to the novice user.

To measure the robustness of a tool in the context of a class, we propose to measure the time between hard and soft failures during the operation of the tool by a person external to the project. We define a hard failure as a failure which leads to the loss of data (e.g. tool crash, system crash, etc.). By soft failure, we mean any behavior of the tool which confuses the user, either because of a faulty behavior of the tool, or a faulty behavior of the user (e.g. non-reported errors, misunderstanding of the documentation by the user, etc.). Every failure should be counted, whether the tool was responsible or not (e.g. kernel crashes). To approximate as much as possible a class workshop, this evaluation should include the training sessions. The robustness evaluation would have to be repeated if the tool is modified before the course (e.g. the port to a new architecture).

Other less time consuming but also more inaccurate metrics for measuring robustness could be used in early phases of the evaluation, such as

- the evaluation of the software development process (how many designers, how many programmers, is any methodology used, which technical documentation is available, etc.),
- the personpower already invested in the tool, and
- the number of non expert users who already used the tool.

Learning curve evaluation

The Bosch course 91 also showed that the robustness of a tool is not a sufficient criteria. A tool may be robust and still hard to use for novices.

During the robustness evaluation, the tester could realize three designs of increasing size: a canned example which needs only half an hour for the expert to build, a midsize design taking one day, and a full size design taking one week. The ratio of the time required by the external tester and the time needed by the expert user for the same three designs would be an indication of the learning curve associated with the tool. If the background of the tester is representative of the background of the participants, this time would also be an indication of how much training time should be invested before the participants are able to realize a design on their own.

Additional remark

The solutions we proposed in this subsection will need significantly more personpower than we spent on tool preparation. However, the robustness and learning curve evaluation would be beneficial for research projects involving systems of tools, as they would give some insight on how well those system would perform in an industrial context. These evaluations could be conducted independently of whether the tool will be used for a specific class or not, thus, the results of the evaluation would be available at the start of the tool preparation.

4.5.2 Long term solutions

This subsection considers long term solutions that would bring into the development phase robustness, ease of use and integration concerns. These solutions are relevant only if these concerns become a major interest of a project (which does not necessary have to be the case).

On the purpose of software prototypes

The purpose of building software in the context of a research institution may be diverse. One can write a software prototype to verify the complexity of an algorithm, show its feasibility, test its integration into an existing system or investigate its fitness to industrial applications. The type of effort which is invested to meet these purposes is also different. Verifying properties of an algorithm only requires an implementation that will be used by its developer for a limited time. Testing the feasibility of a concept does not require a robust prototype. However, estimating how well a new feature might be integrated into an existing system requires the original system to be both robust and well structured. The evaluation of the usability of a system not only requires the system to be robust, but also to be well documented and to have a well thought out user interface.

Early software developed at EDRC was intended for testing feasibility and efficiency. As different prototypes were put together into systems, the average lifetime of a piece of software became longer than a doctorate degree, although the robustness of the systems is comparable to the early prototypes. As a few projects start investigating usability issues (e.g. ASCEND, nDim), and some systems are used in classes such as the Bosch course 91, the purpose of software prototypes in EDRC changed to the point that the way software prototypes are written should change as well. In other words, we claim that converting a prototype intended only for the developers' use into a robust version for evaluation purposes is not realistic.

The next subsections propose a few solutions intended for building more robust tools while not spending personpower required to build commercial quality software.

On the costs of robust research and commercial software

One could argue that developing robust research software becomes comparable to developing commercial quality software, which would deviate from the original goal of EDRC. We do not think this would be the case.

The first concern of a software company is to develop software which can be sold, which does not necessarily mean usable software. From our point of view, commercial quality software includes countless bells and whistles intended to impress the potential buyers, who are usually not the future users of the software. The next few paragraphs illustrate this point of view.

Commercial quality software is often able to load and store data in multiple formats, either to preserve backward compatibility with a previous release or to be able to use data produced by the competition. Research software does not need more than one or two input and output formats, especially if different research projects agree on the same formats, thus decreasing the number of parsers to be written. A commercial tool will also include printing facilities, allowing the user to print out data in various scales and format, which is not essential to research software.

Commercial quality software has a highly customizable user interface, in order to please the largest number of people. The user interface would also include graphic buttons with a 3D relief. Although making a user interface customizable may increase the usability of a tool, we do not think there is a need for providing the users of research software with as many liberties (e.g. on Macintosh, it is possible to set the blinking speed of the caret).

Commercial quality software is sometimes driven by non-technical concerns, such as the required use of a perceived standard. For example, motif is often referred to as an industrial standard, however its interface is not yet robust. Since developers of research software are not interested in selling a product, they have a much broader spectrum of choices, including the use of locally developed software.

The development of robust software is costly, whether it is research or commercial quality software. However, we claim that the development of robust research software can be made affordable if the development effort is concentrated on the essence of the software, rather than on various bells and whistles which find their use only in short demos.

Where tools broke

Research software we investigated during the Bosch course 91 had common weaknesses. Those weaknesses are typical of software prototypes intended for the use of its developer only, but which prevents their use by novice users.

- *Data storage/retrieval.* The storage and retrieval portion of a tool is usually one of the weakest points. The storage/retrieval functionality is implemented near the birth of the tool and is upgraded every time the internal data structure of the tool is modified. Since the kernel of the tool receives more attention at the beginning of the development process, the storage/retrieval portion is more quickly implemented and not well tested.
- *Invocation sequence.* No two tools have the same invocation sequence. Input and configuration files are sometimes specified at invocation, sometimes interactively, and otherwise, the tool assumes a specific file name located in a specific directory.
- *Interprocess communication.* The interprocess communication portion of a tool (when applicable) presents the same types of problems as the storage/retrieval portion: changing formats, lack of testing, etc. To these problems add concurrency due to distributed processing.
- *User interface.* The user interface of research software is usually the one implemented by the developer for testing purposes. It usually requires the user to provide a sequence of low level commands to complete one operation. Constraints and data are expressed in a general purpose programming language. The interface tolerates few errors from the user and does not report any errors.
- *Lack of documentation.* Research software is usually not intended for use by external personnel, which explains the absence of user documentation. However, tools also lack technical documentation, such as file formats and interprocess communication protocol specifications, which makes it hard to integrate any pair of tools together.

Coordination between projects

Some of the problems related to tool integration are not caused by a faulty implementation, but by the diversity of the implementations across projects. Introducing some coordination at the software level between projects would solve most of these problems. By coordination, we mean one person per project (or even possibly one person external to all the projects, such as a programmer, see next paragraph) meeting regularly in order to maximize the reuse of code and experience.

Such coordination would allow sharing a single set of conventions across project for such low level issues as how input and configuration files should be specified, which format to use or for higher level issues like documentation. This set of conventions would apply to the problems usually encountered during the development of a tool, which have no best solutions but many good solutions. Sharing the same solution would possibly reduce any future tool integration effort, and even allow code sharing.

Coordination among projects would also allow the use of a set of standards such as which toolkit to use for building a user interface, or which communication facilities to use for building distributed applications and interprocess communication mechanisms. For two projects which do not conduct research in either of those areas, it does not make much sense to use two different sets of libraries. Moreover, this would allow sharing the purchase and maintenance cost in case of commercial software, thus allowing smaller projects to afford more robust and better documented code.

Finally, the communication necessary for the coordination among different projects would encourage more technical documentation to be written and updated.

Task assignment and training

Another class of problems generally encountered with research software cannot be solved only with coordination. It can be solved only by changing the way software is developed.

Tools developed at EDRC often involved only one student at a time who would devote part of their time to implementation. In some cases, the work of a student would be built on top of the previous work of another student. This process usually results in relatively large and complex systems which are unfortunately only usable by the last developer.

The main cause of the lack of basic software engineering practices is that most students are not software engineers and do not have a formal training in writing large systems. A few different solutions exist to that problem, unfortunately all of them cost additional resources.

The simplest one would be to limit the size of the tools developed at EDRC to a size manageable by a developer with no formal training in software development. Systems would then be composed of a larger number of smaller and more independent tools. In this case, the importance of considering integration issues mentioned in the previous paragraph increases. This solution would still require technical documentation.

Another solution would be to rearrange separate programming and research tasks, by, for example, having Ph.D. students doing research and developing early software prototypes and master

students focusing more on a robust implementation used for educational purposes or usability evaluation. The implementation of a robust system could then be part of a formal training in software engineering. This solution would unfortunately result in longer masters project.

Yet another possible solution would be to increase the number of persons with software training on each project. This solution unfortunately would increase the programmers/student ratio. Moreover, we do not believe that adding qualified personpower at the project level would significantly increase the software quality. Instead, qualified personpower could be independent of any project and focus more on the coordination among projects mentioned earlier. Occasionally, such personpower could also be used for the robust implementation of general purpose software which would be either too expensive to purchase or not available (e.g. DPSK[1]). Such development effort by qualified personpower should be limited in order not to disperse the resources. We would find it more useful to have such personpower available for coordination, consulting, or even formal training.

Additional remark

None of the solutions presented in the previous paragraph could solve the software quality problem alone and for free. However, in cases where the quality of the software prototype is of interest, we do believe that a combination of those solutions along with the awareness that the implementation of a large system is a complex design process itself can significantly increase the benefits of the software development effort.

1. DPSK is a package developed at EDRC providing functionality for distributed software.

5 Teaching

5.1 Overview

This section describes the educational aspect of the project. First the schedule and the content of the project lectures, along with the preparation time are described. Next the use of various visual aids for presenting tools to the participants are discussed. The remainder of the section identifies the problems encountered during the project lectures and proses a few alternate solutions.

The data presented in this section comes from the handouts of the lectures, e-mail messages, and interviews conducted with four of the participants at the end of the course.

5.2 Project organization

All members of the project team participated in the preparation and the teaching of the project lectures. The tool providers conducted a workshop or a demo of their tools. The members of the project design team presented the part of the artifact they designed. The faculty member taught the lectures on product concept, digital design and software engineering.

Figure 17 shows the planned and actual schedule of the project. The vertical axis represents time. The gray background represents the two six week segments of the course. Every rectangle corresponds to a topic. The rectangles with a gray outline represent topics which were planned but not completed. The number in the rectangle is the number of afternoons spent on that topic. For the topics which were not completed, the number in the rectangle is the number of planned afternoons.

All the participants went through the introduction to the computing facilities, the product concept and the electronic specification. The class was then split into a novice track and an expert track. The left column represents the topics seen only in the novice track. The rectangles besides the rectangles in the novice track are topics only seen by the expert track. The next few paragraphs describe each topic individually.

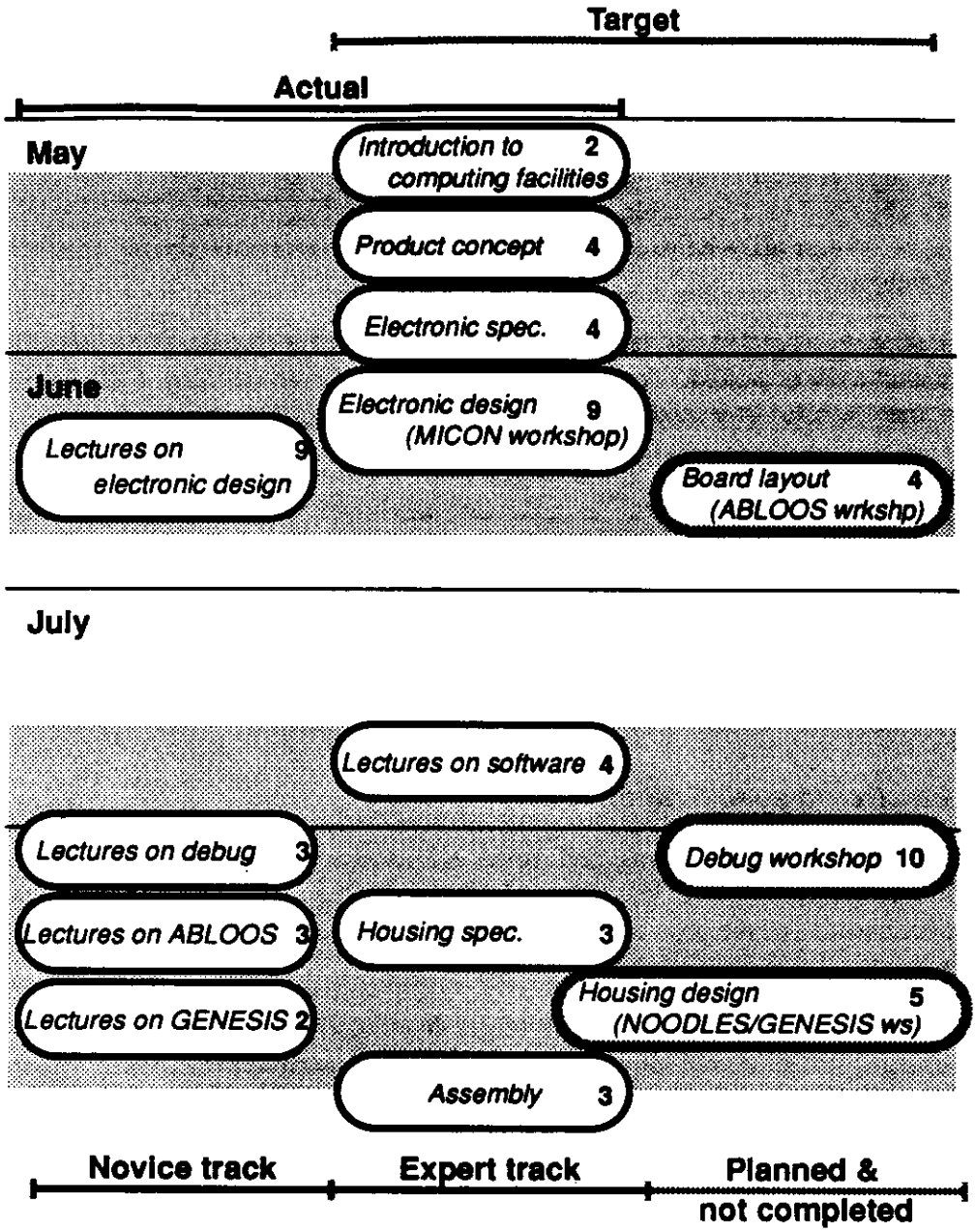


Figure 17 Project schedule

'Setup' week

The first week of the project consisted of a three day lecture on group management and organization given by an external consulting company, and a two day introduction to the local computing facilities. Following the 'setup' week, the participants were divided into four groups taking into account their strengths and weaknesses in mechanical, electrical and software engineering.

Product concept

The first problem occurred when we handed out the specification of the artifact. Some of the participants were involved in a similar project in their company. They did not want to work on the design of a map display in the context of a general course claiming that any idea they had on this subject was the intellectual property of the company they were working for.

Since much time and resources had already been invested in the design of the artifact, the specification was changed to a portable computer to display blueprints, targeting construction workers. This slight change allowed us not to loose the time and resources already invested, while satisfying the participants.

Once the specification was handed out, teams were allowed to ask for clarifications during a twenty minute question and answer period. They were given three days to develop a product concept. After three days, they handed in a product proposal and gave a presentation.

The product concepts handed in by the teams ranged from a $700 three button pocket computer to a $10,000 PC compatible computer with an extended keyboard. All the proposals used 25x80 character LCD displays.

The first managerial decision consisted of giving the teams tight cost ($400, including $150 for the display) and weight (1kg) constraints. The main purpose of these constraints was to bring the participants back to a product which could be designed and manufactured within the duration of the course. We also introduced the Private Eye display at this point.

Electronic specification

Given the above mentioned constraints and some additional performance requirement ("the prototype should be able to display at least three blue prints"), the teams had one week to develop the electronic specifications of the product. By electronic specification, we meant processor family, clock frequency, memory sizes and external interfaces.

The electronic specifications handed in by the teams happened to be similar, due to the small proportion of experts in electrical engineering among the participants (see Figure 2). We discovered that the teams would collaborate closely when they lacked information or experience. All the teams proposed the use of the 8051 microcontroller at a clock frequency ranging from 5 to 12 MHz, a RAM of 2KBytes and a ROM of 4KBytes. In all designs, the blueprints were stored in removable cartridges.

The reaction of the participants to the first set of constraints (after the product concept and before the electronic specification) was frustration. Although they understood that the purpose of the constraints was to keep the design simple, they had the impression they were designing a toy artifact and were far from simulating a realistic design process. Along with the specification of the golden solution, we presented the characteristics of two industrial designs of artifacts which would have met the performance and cost requirements of the specification: the Nintendo Entertainment System and the Commodore 64.

Lectures on digital design

The small proportion of participants with electrical engineering experience, and the weak computer skills observed during the introduction to computing facilities, led us to split the participants into a novice track and an expert track. The expert track consisted of the participants interested in completing the detailed design using the CAD tools. During the expert track lab sessions, the novice track had lectures on digital design. The goal of the lectures was to provide the novice track with enough knowledge of digital design to understand the resulting work of the expert track.

The lectures on digital design were a condensed version of a required junior computer engineering course taught at Carnegie Mellon (*18-247: "Introduction to Computer Architecture"*). The novice track introduced basic concepts in computer architecture, memory hierarchies, I/O subsystems and basic concepts in reliability. The lectures occupied the last two weeks of the first half of the course.

Electronic design

The expert track were trained during one week on the MICON tool suite, and spend another week designing the electronic portion of the artifact. All teams went through the major steps of a design process with MICON, and one team actually completed a design.

The expert track suffered two major system crashes and about four other less serious failures during the use of the tools. The unexpected frequency of system failures, as well as the high sensitivity of the participants to those crashes, led us to diverge from the original plan and to decrease the involvement of the participants in computer aided design. The specifics of the system failures were discussed in Section 4.4.

Board layout

A lecture on ABLOOS and general layout problems was given towards the end of the first six week segment of the course. The data structures and problem solving policies used by ABLOOS were introduced.

The initial plan of the project also included an ABLOOS workshop, in which the participants would place and route the electronic design produced by MICON. The participants would have been given a crude solution produced by ABLOOS with a minimum of constraints. Their tasks would then have been to improve the quality of the layout and shorten the CPU time needed to find it by adding constraints to the inputs to ABLOOS.

After the various system failures encountered during the MICON workshop, it was decided to replace the ABLOOS workshop by a series of lectures and demonstrations of the tool. The main motivation of this decision was that we considered MICON as more robust than ABLOOS.

The lectures were given in parallel with an on-line demonstration which was projected on a white screen in the class room. Instead of modifying the input files to ABLOOS themselves, the participants would suggest additional constraints to add on the input file.

Lectures on software

Four afternoons were spent on teaching software. The first two lectures were an introduction to software engineering and the major issues involved in the cost and development of software in general. The following lecture was spent discussing the program which was being developed for the blueprint display.

Lectures on debug

The initial schedule planned a debug workshop in which the participants would populate and test part of the board for the blueprint display. At this point of the course, the main printed circuit board was not manufactured yet and not tested. The debug workshop was replaced by lectures which presented debugging procedures and the detailed design of the printed circuit board. The problems encountered during the debugging of the first prototype (i.e. the cause of the delay in the manufacturing) were taken as an actual example (see unforeseen problems in Section 3.5). A commercial tool for routing was also presented during one lecture.

Housing specification and design

The teams were given a set of mechanical constraints and the actual dimensions of the printed circuit boards. Following a course on industrial design which emphasized ergonomics, the teams were asked to realize two prototypes each, of housings using styrofoam and cardboard.

No golden solution was presented for this phase. The design used in the manufacture of the prototype was a refinement of a design submitted by the participants.

The NOODLES and GENESIS workshop was replaced by lectures on solid modeling and solid generation, as well as a demo of each tool.

5.3 Visual aids

Various visual aids have been used with different levels of success in presenting the tools to the participants. The following list defines the names referring to visual aids we used in the remainder of this document:

- *Screen dump:* a printed image of the workstation screen shown on an overhead transparency
- *Screen projection:* the screen of a workstation projected on a large white screen in the front of the classroom
- *Shared window:* the screen of a workstation replicated in a window on multiple other workstations
- *Demo tape:* a video tape played in the classroom
- *Demo in small groups:* a demo on a single workstation shown to groups of five to eight persons

Screen dumps were useful during lectures where a suite of tools was presented. They allowed us to focus on the important points of the tools and hide irrelevant details which could distract the participants (e.g. translator invocation, system crashes, etc.). They also allowed a more compact presentation by removing the overhead of tool invocation. When a copy of the screen dumps was handed out before the class, the participants also find it useful to be able to take notes directly on the screen dump, thus serving as a permanent storage[1].

Screen projection did not always meet our expectations. The screen projector used in the computer cluster was not intended for such large rooms, and often presented focusing problems for color screens. In other cases (e.g. Viewlogic), the font used by the tool and the dimensions of the drawings were too small to be seen by all the participants. We found screen projection useful only for groups of about ten persons for black and white displays.

Shared windows were used as a replacement of screen projection. Software which would replicate the screen of the instructors workstation in a window on the participants workstations was used. The refresh rate was too slow (about once every three seconds with a latency up to five seconds) for this visual aid to be useful. Another problem was that the cursor information was not propagated to the participants screen, thus making it difficult to point at a specific object on the screen. Nevertheless, this technology seems to be promising once the performance problems were solved (e.g. sharedX on HP700).

A demo tape was used during the first lecture of the project for introducing the tools. As with screen dumps, a demo tape was useful for giving an overview of many different tools in a short time. It also give the participants an insight on how those tools could be integrated into a design process.

1. This fact was mentioned by some participants during an interview. They would use a screen dump to record the terminology used in various displays and windows of the tools, and to write down keystroke sequences. However, some participants claimed screen dumps were not useful at all.

5.4 Towards better solutions

This section focuses on the educational aspects of the projects which could be improved. Most of the suggestions in this section come from the results of a survey of the participants which took place at the end of the course. Two participants from each track were interviewed by groups of two for one hour. Questions on the content, on the structure of the project, on the use of tools and on the use of visual aids were asked. The results of the survey on visual aids have been presented in Section 5.3.

Project content

All the interviewed participants expressed that they were not equally interested in all phases of the design process. They would have preferred to spend more time on tools and methodologies related to their field. They found it difficult to follow lectures and workshops with people of such diverse levels and background (see Figure 2). They suggested classes should be organized with a more homogenous set of participants, either by splitting the participants into different tracks (software, electrical and mechanical engineering), or by organizing a course with a more focus content.

The participants also expressed their interest in a more in-depth training on each tool, i.e. they would have preferred longer and more focused workshops at the cost of seeing fewer tools. They suggested workshops should only be given on tools which were not available in the industry, and not include any commercial or traditional tool.

The opinions expressed about the content of the project itself were diverse. Some participants liked the idea of having a single global project theme (i.e. designing an artifact). One of the participants did not find it useful and too ambitious. All of them would have preferred more freedom in designing and manufacturing their own artifact, at the risk of realizing a nonworking design.

Project structure

All the interviewed participants found the course too long and too intense to be fully efficient. They found themselves not being able to assimilate any new material after the eighth or ninth week. They suggested more free time for reviewing course material, introducing homework and graded tests. They promoted tests as being useful for providing feedback to both the instructors and the participants.

In the context of the project, one participant suggested to provide the participants with their own workstation for the full duration of the course. He believes the participants should spend more time using the tools by themselves.

The interviewed participants suggested the following workshop scenario (for a single tool):

- *General overview (one afternoon):* its purpose, its context and its limitations are presented; user documentation is handed out
- *Hands-on workshop (one afternoon):* the low level details of the tool are introduced to the participants by means of a very simple example; the lecture alternates between course mode and workshop mode; the specification for a midsize design is handed out

- *Midsize design (two afternoons):* the participants are on their own and built a midsize example; TAs are available for questions and assistance only; designs are handed in at the end of the second afternoon
- *Critique (one afternoon):* the designs of the participants are graded and critiqued

One participant particularly insisted that the content of the class should be structured and exposed in a top down fashion. For example, he did not like the overlap between the system level training and the tool workshops.

Two participants found that the most successful classes were those which were based on a very simple and working example. They found the worst classes to be those where time was lost because of computer crashes. They pointed out that they were less able to recover from system crashes than students, mainly because they were accustomed to reliable systems.

Finally, a participant pointed out the importance of the overall coherence of the project material. They found the demonstrations which were presented out of context were not useful. This advocates for a global project theme.

Tools

The participants found that most of the tools seen in the workshop suffered from a data management problem. They spent a non-negligeable part of their time browsing, renaming and moving files. They said the lack of transparency of the tools with respect to the file system would require a more in-depth introduction to the file system. On the other hand, tools with a better interface would decrease this learning time and allow more focused training.

Additional remark

Most of the suggestions of the participants would require more teaching assistants and resources. Considering the time to train such personpower and the setup cost associated with a computer cluster, we do not find making the course shorter and more focused a realistic solution. In order to further lower the overhead, we would even suggest to reuse the same artifact and large portions of the project material across the courses. This would be reasonable since it is not likely that the same participants will attend such a course two consecutive years.

6 Towards a better project organization

6.1 Overview

A simplistic solution to many problems we previously mentioned could have been to add time and personpower to the project. However, larger personpower and longer preparation time mean higher cost, which would not have been acceptable. In this section, we focus on the problem: given a fixed amount of resources, how productivity could have been improved, and contingencies and overhead reduced.

6.2 Task assignment

As mentioned in Section 3.4 and Section 4.3, the calendar time of each task is usually greater than its personpower; in other words, a task could take one calendar month and only one person week. The reason is that each person was responsible for more than one task at any point in time. The TAs were responsible for the design of the artifact, teaching and administering the project. The tool preparation also overlapped with the beginning of the course. This resulted in lower productivity and shorter deadlines.

Increasing productivity (while keeping the personpower constant) could have been achieved by increasing either the size of the team or the preparation time. The former would have enabled the assignment of only one task per person. The latter would have allowed the tool preparation, the design of the artifact and the teaching to be sequential. This solution would be better because of its lower overhead in training.

In the context of the Bosch course 91, none of those solution would have been possible. The preparation only started in March, when the number of participants (hence the budget of the course) was known, and the project team started to work full time by May, when the academic semester ended. We did plan to increase personpower by training three REUs. This happened not to be a realistic solution because of the high training overhead.

6.3 Relationship with the rest of the course

Another consequence of the late start in the preparation of the course was that the material taught in the morning lectures was not coordinated with the project schedule. This lack of integration often required presentation of the same background material twice. A better integrated schedule would have allowed for a more fruitful project execution and participation.

6.4 Interaction between design processes

Figure 5 illustrated the interaction between the design process of the project team and the design processes of the participants. Initially, the design of the first artifact was planned to be completed and tested before the beginning of the course. In this scenario, the first artifact would have been used as a golden solution that the participants would manufacture instead of their designs. The project team design process was delayed and overlapped the participants design process, to the point that the housings designed by the participants served as a starting point for the project design team.

Both the quality of the artifact and the interest of the participants could be increased if the participants have some influence in the design of the artifact. If the project team design process is completed before the beginning of the course, and in the case off campus manufacturing times are short and predictable, the ideas of the participants could be used to improve the initial design. The trade-off is that the project design team has to be available during the course and able to modify the initial design without introducing too many flaws.

6.5 Tool providers

Most of the work on the tools was done by the software TA, since the tool providers were graduating. The first consequence was that the latency of bug fixes was high. The second was that the most recent version of the tool was unstable, because it had new features added.

Having a person external to a project do low level software tasks such as porting and debugging induces a high overhead in training. Moreover, having a single person work on many tools requires a broad software background (knowledge of multiple programming languages, architectures, databases, user interface toolkits, etc.), which was not always the case for the software TA.

A better solution would have been to have only the tool provider modify the code of their tool. This suppresses any overhead in training. However, the trade-off is that the tool provider has to be available for an extended period of time, and that he has enough programming skills to be able to port a tool to a different architecture and a different display.

7 Conclusion

In this document, the preparation and implementation of the project part of the Bosch course 91 was described and analyzed. We suggested a few solutions we believe would improve the quality and decrease the cost of a similar course. The relationship of such a course with research was also investigated.

We focused on the importance of the estimation of the personpower and the calendar time required for such a course. We described how delays in the preparation could significantly increase the cost and lower the teaching efficiency.

How the organization of the team project and the task assignment could be improved in order to increase productivity was discussed. We came to the conclusion that a larger team would not solve the personpower problem encountered during the course; that a team of similar size with a longer calendar preparation time would be preferable.

An insight was given into the cost of integrating a tool into a course program and suggest that a smaller number of better quality tools would be preferable. This would lower the preparation time and allow a more in-depth training of the participants.

We made long term suggestion to improve the quality of research software without turning EDRC into a software factory. We believe that if tools are to be used in a teaching environment, robustness becomes a major concern which has to be addressed early in the development phase of the software.

Finally the participants' side of the course was presented thereby illustrating their expectations in terms of quality and course content.

8 Bibliography

[Birmingham et al, 1989a] W. P. Birmingham, A. P. Gupta and D. P. Siewiorek
The MICON System for Computer Design
IEEE Micro, Vol. 9, No. 5, October 1989

[Birmingham et al, 1989b] W. P. Birmingham and D. P. Siewiorek
Capturing Designer Expertise -- The CGEN System
26th ACM/IEEE Design Automation Conference Proceedings, 1989

[Coyne, 1991] R. F. Coyne
An Evolving Hierarchical Design Framework
Ph.D. Thesis, Carnegie Mellon University, EDRC
Technical Report EDRC-02-15-91, May 1991

[Gupta et al, 1990] A. P. Gupta and D. P. Siewiorek
M1: A Small Computer System Synthesis Tool
6th IEEE Conference on AI Applications Proceedings, 1990

[Gursoz, 1991] E. L. Gursoz
User's Manual for Noodles Library
Carnegie Mellon University, EDRC
May 1991

[Gursoz et al, 1991] E. L. Gursoz, Y. Choi and F. B. Prinz
Boolean Set Operations on Non-Manifold Boundary Representation Objects
CAD, Vol. 23, No. 1, January 1991

[Heisserman, 1991a] J. A. Heisserman
Generative Geometric Design and Boundary Solid Grammars
Ph.D. Thesis, Carnegie Mellon University, EDRC
Technical Report EDRC-02-18-91, May 1991

[Heisserman, 1991b] J. A. Heisserman
Genesis Reference Manual
Carnegie Mellon University, EDRC
Technical Report EDRC-48-23-91, August 1991

[Subrahmanian et al, 1989] E. Subrahmanian, A. Westerberg, G. Podnar
Towards a Shared Computational Support Environment for Engineering Design
Lecture Notes in Computer Science: Collaborative Product Development, D. Sriram, R. Logcher, S. Fukuda (eds)
Springer-Verlag, 1991