The Groupthink Specification Exercise

Michael D. Ernst

MIT Computer Science & Artificial Intelligence Lab
Cambridge, MA, USA
mernst@csail.mit.edu
http://pag.csail.mit.edu/~mernst/

Abstract. Teaching students to read and write specifications is difficult. It is even more difficult to motivate specifications — to convince students of the value of specifications and make students eager to use them. The Groupthink specification exercise aims to fulfill all these goals. Groupthink is a fun group activity, in the style of a game show, that teaches students about teamwork, communication, and specifications. This exercise teaches students how difficult it is to write an effective specification (determining what needs to be specified, making the choices, and capturing those choices), techniques for getting them right, and criteria for evaluating them. It also gives students practice in doing so, in a fun environment that is conducive to learning. Specifications are used not as an end in themselves, but as a means to solving realistic problems that involve understanding system behavior.

Students enjoy the activity, and it improves their ability to read and write specifications. The two-hour, low-prep activity is self-contained, scales from classes of ten to hundreds of students, and can be split into 2 one-hour sessions or integrated into an existing curriculum. It is freely available from the author (mernst@csail.mit.edu), complete with lecture slides, handouts, a scoring spreadsheet, and optional software. Instructors outside MIT have successfully used the materials.

1 Introduction

Specification (along with related verification activities such as testing) is critical to the success of any real software system. However, many students view specification as a dry, tedious, and impractical topic.

One problem is that many undergraduate programming classes fail to integrate specification into the curriculum in a realistic way. Writing a specification for the purpose of being graded takes specifications out of the context in which they are used, so students gain little appreciation for their utility. Additionally, typical class assignments are simple, so techniques that are crucial for more complex software may not be cost-effective. Students learn the (incorrect!) lesson that specification is pointless.

This paper describes Groupthink, a two-hour group activity that teaches students about specification — conveying its utility, illustrating how to do it (and how not to do it), and emphasizing the importance of teamwork and communication during the process — through hands-on experience with a realistic problem.

P. Inverardi and M. Jazayeri (Eds.): ICSE 2005 Education Track, LNCS 4309, pp. 89–107, 2006.

In small teams, students are given a set of requirements and asked to specify the behavior of a simple desktop telephone with integrated answering machine. They then play a game to determine how successful their specification was. The game poses questions about the behavior of the telephone, and each team member must answer the questions individually. There are no right or wrong answers; rather, the goal is for all team members to give the same answer, indicating they have constructed a specification that they can all understand and that covers all possible behaviors. Finally (though they are not told this beforehand), they are given a second chance to improve their specification and to continue to play the game.

The author has run the activity 9 times since 2002, involving over 850 sophomores from the MIT School of Engineering. The activity has also been run outside MIT. At MIT, the activity is part of the January inter-session UPOP (Undergraduate Practice Opportunities Program) engineering "boot camp". This is an intensive week-long program intended to teach engineering sophomores a variety of engineering and management skills that are often not taught in regular classes but are important for industrial jobs [9].

The objective of the exercise is to teach students

– the importance of creating precise, complete specifications
– the difficulty of doing so
– specific methods for eliciting requirements and creating specifications
– strategies for information representation when expressing specifications
– effective teamwork and group communication
– the importance of ensuring understanding among all team members
– time management
– the value of iteration and how to learn from mistakes

No brief activity can teach all these skills, and different students bring different skill sets to the activity. Therefore, the exercise is intended to teach different lessons to different students. As explained in Section 4, the exercise meets its objectives. Students find the activity enjoyable, they report learning these lessons, and they demonstrate better success on a specification task afterward.

Our three major goals in designing the activity were as follows.

1. We wished to pose students a well-motivated, practical, concrete, realistic problem. Such a problem is not a dry academic exercise, but has a clear connection to real-word work. The problem needed to be small enough to be easily comprehensible and to look easy at first glance, but hard enough that undisciplined approaches were unlikely to be successful, so that the students appreciate the advantages of using a specification.
2. We wanted to run an interactive and lively event. Many people learn best by doing (e.g., active learning [1], which is promoted by a variety of teaching guides [3,10,11,2]), and hands-on activities are more likely to be fun and to inspire students. Iteration gives students an opportunity to learn from and to correct their mistakes, gaining a feeling of accomplishment and illustrating how and why undisciplined alternative approaches fail. Interaction with other

people exposes students to multiple technical and organizational approaches to a problem. An element of competition — both with other teams and with a team's own previous performance — made students more energetic and focused them on the task.

3. We aimed to appeal to students with many different backgrounds. The activity should equally engage computer science students and computer-phobes, and those with years of job experience or none at all. This implies that the activity must teach different skills and concepts to different people, and it should encourage participation by all students. For example, we wished to force "techies" to talk to "non-techies", which teaches better communication skills to both groups.

2 The Activity

The activity starts with a very brief lecture on specifications. The introductory lecture motivates why one would care about specifications, and who should care about specifications. Writing and understanding specifications is important to everyone:

– designers – to communicate their ideas to others
– implementers – to faithfully reproduce the ideas
– testers – to ensure compliance
– managers – to understand what the technical team is doing
– technical support – to understand what behaviors are desired
– salesmen/marketing – to communicate to customers
– users – they cannot use, and will not buy, what they cannot understand

Students then form small teams and are given a partial specification of a system, plus a set of requirements for its behavior. The specification is incomplete and inconsistent in somewhat subtle ways. Each team is asked to develop a better and complete specification that prescribes the system's behavior under all possible user interactions. There is no right or wrong specification as long as the requirements are met. The goal is for each team to agree upon the specification and to understand what they have agreed upon. The exercise is motivated in part by imagining that the students are working for different divisions of a company that wants to bring the system to market.

The Groupthink framework provides a way of running an activity that emphasizes teamwork and consensus. Instructors use Groupthink along with a content module that sets students a specific task. As of this writing, one content module, called "Answerphone", is distributed with the Groupthink Specification Exercise. Appendix A reproduces the requirements handout that is given to students.

Answerphone is a desktop telephone with integrated answering machine. Other systems would also work well. The system should be relatively familiar to students to minimize confusion and reduce time requirements, and should have a moderate level of feature interactions to make the specification rich but tractable.

Each team of students had 30 minutes to agree upon the system's behavior. After 20 minutes, facilitators such as instructors or TAs (teaching assistants) may gently guide teams that are struggling very badly: for instance, some students are not engaged, so they aren't participating, or some students don't understand the team's solution. TA intervention should be rare, because we found that students learn best when they struggle, make their own mistakes, and then correct them. Thus, TA involvement is primarily intended to let the team proceed to new experiences and mistakes that can teach new lessons. TAs should avoid making specific suggestions, but might point out that all team members must understand the solution, or note that the team is stuck on one relatively minor point, or ask whether the team has considered some important situation. The TAs should listen to all of the group's deliberations, in order to provide better feedback after the entire activity was over.

To evaluate their specifications, students play the Groupthink game that gives the activity its name. Teams are asked a series of questions about the system behavior, many of which focus on feature interactions. Each question is displayed on an overhead projector and is also read by the instructor. A sample question is:

> The user is connected to an outside party. The outside party hangs up.
> Can the user hear dial tone?
> A. The user hears dialtone (the phoneline is in the lineactive state)
> B. The user does not hear dialtone (the phoneline is in the lineidle state)

Each team member individually answers the questions. There is no right answer. Instead, the challenge is for all team members to give the same answer, without communicating with one another, based on the specification they have developed. It is important to emphasize that differences in answers are a failure of the team to develop a specification that everyone understands, not a failure of any individual.

Displaying a running tally of scores promotes excitement and involvement in the game. Each team receives as many points as the size of their plurality answer, plus a bonus of 10 points if all answers agree. For example, suppose that a team has 9 members. If 4 team members answer "A", 3 answer "B", and 2 answer "C", the team receives 4 points. If 1 team member answers "A", 1 answers "B", and 7 answer "C", the team receives 7 points. If all 9 team members answer "B", the team receives 19 points (9 points plus 10 bonus points). Students quickly realize that winning or losing the game depends primarily on the number of bonuses earned.

After a round of questions, the instructor moderates a class discussion that conveys the core of the material the students will learn. Students explain what strategies were and were not successful. We believe that students hearing from one another in their own words is more effective than a lecture: it is more convincing to hear from someone who has actually tried a particular technique. This approach also forces reflection (introspection), which is itself a successful learning strategy [6,2,12]. An instructor moderates the discussion, following up to emphasize or expand upon certain points, or to draw connections. Section 3 lists some common student observations.

After the discussion, though they are not told this beforehand, the students are given more time to improve their specification before a second round of the game (see Section 2.1). Students use their mistakes from the first round to help them improve. This iterative learning experience is key: students get to make mistakes, learn from them, and then do a better job, which solidifies the learning and leaves them with a positive feeling [16].

After the game, the activity concludes with a lecture that reinforces the points made by the introductory lecture, the class discussion, and the exercise.

2.1 Round 2 of the Game

The second round of the game — after students have played the first round, then improved their specifications based on their experiences — makes four scoring-related changes.

Two changes keep all teams involved in the game. These changes prevent any team from getting so far ahead in the first round as to discourage other teams.

- All point values are doubled (including bonus points; just double the overall score).
- There are two winning teams: the one with the highest overall score and the most improved team.

Two other changes require students to produce a quality specification. These changes make the second round more challenging, and they prevent participants from "gaming the system" by creating an unrealistic system that is easy to answer questions about.

- Some answers are marked as wrong, because the behavior violates requirements or the operation of the telephone network. When scoring, we disregarded wrong answers when determining the plurality answer.
- Some answers are marked as legal but questionable — for instance, they do not provide the functionality that users would desire. If such an answer is the plurality, the team's score is not doubled.

Before introducing new questions, the second round repeats all the questions that the first round asked. (Students are *not* appraised of this beforehand.) After the repeated questions, the instructor should ask whether any teams missed any of the questions, ask why, and emphasize the importance of learning from one's mistakes. Then, continue with the new questions.

3 Lessons for Students

During the class discussion, students commonly report having learned the following lessons. Instructors may wish to emphasize these points, or to raise issues that they observed but that no student voices.

Student remarks tend to fall into two categories: technical approaches and team organization. Some common technical approaches are:

- Draw state diagrams with transitions among states.
- Draw state diagrams showing the state of all components (switches, phone-line, etc.).
- Explore a set of "use cases" or scenarios, and decide what happens in each circumstance. This was effective when the participants predicted the questions, but less so when they did not.
- Draw a flow chart indicating the sequence of events.
- Focus the specification around the communication protocol among system components.
- Rank all actions: give some priority over others. An extension of this is to create a rule-based system with a set of (prioritized) rules.
- Decompose behaviors into normal and exceptional cases, and treat each separately.

Common approaches to team organization include:

- Choose a leader to direct and focus the group's activity.
- Don't argue. To avoid wasting time, poll everyone, make a decision, and then move on.
- Choose simple behavior that is easier for everyone to understand; avoid bells and whistles. There is no need to add complicated features that are not explicitly noted in the requirements, such as call waiting.
- Make a schedule indicating how much time it is worth spending on each task, based on the value of each discussion or decision, and stick to the schedule.
- Listen to everyone — don't shut out some participants. Often quiet team members had very good points to make, and those with more forceful personalities need to listen carefully.
- Ensure that everyone understands — the team is only as strong as its weakest member. Team members comfortable with the technical terms used in the requirements should help the others. Students come to realize that although the exercise is couched in telecommunications and computing terminology, it does not rely on skills specific to those disciplines.
- Use a master copy so everyone can focus on it. When this is on the whiteboard, more people tend to participate then when it is on paper, and it is more useful than each person writing up a separate copy. If there are separate copies, they must be kept in sync.
- Spend a few minutes in which everyone reads the handouts before diving into a solution, in order to guarantee understanding of the problem.
- Divide and conquer: delegate different parts of the specification to different sub-teams. Re-group in plenty of time to explain the separate sub-teams' progress to the full team.

The instructor should emphasize that the technical aspects and the team organization are complementary, and neither can be ignored. If either is mismanaged, the project will be a failure.

4 Assessment

The Groupthink specification exercise has been a success. Students eagerly and intently attack the problems during the activity; they loudly cheer and groan as their team's fortunes change during the game; and they confirm their enjoyment and learning in a post-activity survey. TAs for the UPOP engineering boot camp [9] report that the activity is their favorite among the more than a dozen presented by engineering and management faculty.

We attribute the success to the design of the activity, which paid careful attention to the goals noted in Section 1. The problem was realistic enough to motivate students, hard enough to require use of specifications, and easy enough to give students a sense of accomplishment. Students got a chance to fail in an initial attempt, then a chance to succeed after reflecting on their experience and learning about a wide variety of other approaches taken by other teams [16]. Achieving high scores required full participation from every team member; this motivated students to communicate and explain their ideas in a simple and clear manner. Students with different backgrounds learned different lessons, whether technical, communication, or organizational.

4.1 Student Learning

This section reports results from four sources: team scores in the activity (collected from the scoring spreadsheets), student surveys, a post-activity followup by a student, and instructor observations of team behavior.

Student performance improved during the game. We compared (normalized) scores during the first round of play (before students had observed their performance and participated in a class discussion about various technical and organizational approaches) and the second round. The improvement ranged from 4% to 131%, with an average of 66%. Student self-assessment indicated that, compared to a control group, the UPOP students had greater improvement in their teamwork abilities [9]. (Before UPOP, 18% of students reported never having worked as part of a team either in class or on a job; 36% reported one such experience; and 46% reported 2 or more such experiences.)

A post-activity survey in 2002 asked students what big ideas they had learned that they could use in practice. The most common themes were clarity (12%), teamwork (12%), communication (11%), simplicity (11%), the importance of a written document (9%), completeness (5%), focus on the end goal (5%), and organization (5%), though students also cited a dozen other general ideas. Many of these themes are related, but the diversity also demonstrates that the activity can teach different ideas to different students.

When asked what activities they had enjoyed best in a session of the MIT UPOP program, student responses included "Answerphone brought out the big idea of communication with every member of the group," and "The Answerphone activity caused us to really gel as a group and communicate very effectively between ALL group members." Other students remarked that they enjoyed the analysis portion, and the friendly competition (see Section 4.3).

The activity turned out to be surprisingly realistic and practical. One student wrote to the organizers of the UPOP program during the summer of 2004:

> I am on my second day of work at my Internship at Orange (division of France Telecom), and I am looking at design specs for a product I am researching — a voice activated answering system. As I opened the packet, I saw a flowchart very similar to what we did in the Answerphone exercise back in UPOP. Good job on picking the activities we did.

Not every student or team learned the lessons presented in Section 3. Based on direct observation of 102 teams, the largest problem was individuals who were not engaged: when it came time to give an answer, those people too often gave a divergent answer, preventing the team from achieving unanimity (and gaining the bonus points). Two common reasons for disengagement were bossy individuals who ignored and alienated other team members, and individuals who had less technical background than others but failed to speak up to ask for clarification. Other common problems were failure to understand the problem domain, losing the big picture (overfocus on small issues, resulting in an end product that was missing large and important components), and poor technical approaches that were not corrected in time. On a few occasions, a team that had excelled during the first round became overconfident and did not use the second specification period effectively, and thus fared worse on the second round of questions.

4.2 Student Reaction

As noted earlier, students and TAs enjoy the Groupthink specification exercise. Students had three general complaints. The first complaint concerns teams of different sizes; see Section 5.3.

The second complaint is that some students tried to game the system. The introductory lecture emphasized that it is not acceptable to communicate during the questions nor to use meta-information, such as answering with the first listed answer in cases of ambiguity; students respected these restrictions. The complaints were rather that some teams had made unrealistically operable specifications that happened to be easy to reason about and answer questions regarding. Since simplicity and understandability are key lessons of the exercise and are crucial in real-world projects, this was not an unreasonable approach. We also addressed this issue in the second round of the game (see Section 2.1) by awarding lower scores for illegal and for strange answers.

The final complaint is that students wanted more time: they felt that they could have done a far better job if given the opportunity to further improve their specifications. We answered this complaint by noting that in real engineering situations, there is never enough time (it is constrained cost and time-to-market concerns), and that time management is one of the crucial skills that the exercise teaches. And though the time is artificially compressed in the exercise, the problem has also been made artificially simple (another point emphasized in the materials), so the time-to-difficulty ratio is not unreasonable.

All of these complaints have become less common as we have fine-tuned the game over time. For example, time management has become a more important theme of the activity over time.

4.3 Experience Without Competition

An instructor at another institution ran the activity without the competitive aspect.

> Each class was incredibly small (about 5 students), which is unusual. Because of that, I grouped all the students in each class into 1 group. The more advanced class set about doing it in a serious manner, while the "Intro to programming" class basically blew it off (since I'd mentioned that there were no points attached). Something I thought was really interesting is that the Intro class (which went second) accidentally saw the scores for the other class only after their first round of specification (which I'd forgotten to delete from the `.xls` file); they commented that had they known it was going to be competitive, they would have actually tried harder. Also interesting is that all 5 of those people were male, and I'd estimate the ages to be 18–26 ish or so.
> I found the exercise to be incredibly well-thought out, and greatly appreciate the handouts / slides / outline — it was incredibly helpful!

5 Mechanics

5.1 Schedule

We ran the exercise as a two-hour self-contained activity (see Section 5.2 for alternatives), following this schedule:

5 min	Brief introductory lecture
10 min	Explanation of requirements
30 min	Specification (round 1)
10 min	Game (round 1)
20 min	Discussion
20 min	Specification (round 2)
15 min	Game (round 2)
5 min	Game wrapup
5 min	Concluding lecture
120 min	Total

We have found that the introductory lecture is most effective when it is shortest. The explanation of requirements walked the students through most aspects of the telephone system and answered their questions. During the class discussion, we started with the teams with the highest and lowest scores, and after that proceeded around the room systematically. We made a representative from each team speak. The game wrapup can include more discussion (we found time to debrief in the individual teams very valuable), awarding of prizes, etc.

5.2 Integration into Curriculum

Because the Groupthink specification exercise is self-contained, it can easily augment an existing curriculum, for instance to motivate specifications or to replace dry lectures. Alternately, by incorporating more instruction about specifications, the problems presented in the game could be made more challenging.

The exercise can be split into two one-hour class sessions. The first session introduces the activity and plays the first round of the game along with limited discussion, and the second session includes the second round of the game, preceded and followed by more discussion. The exercise can also be integrated with more lectures or other instruction, enabling it to fit into an existing curriculum or the problems to be made more challenging.

Variations on the activity as run so far would be interesting. One example is to give students far more time to perform the activity, eliminating any time pressure (or perhaps increasing the complexity of the task). Another example is to perform a similar activity using formal rather than informal specifications.

Instructors can develop additional modules (in addition to the provided "Answerphone" module) by creating new handouts for students and new questions. Section 5.5 describes the voting techniques for the Groupthink specification exercise. When using the manual voting technique, the questions should be in slideshow format (e.g., PowerPoint), and when using the electronic voting technique, the questions should be in an XML format that is documented with the provided software.

5.3 Team Size

We run the activity with up to 13 teams of 9 students. While the team size is in part dictated by the physical setup of our lecture room, this size has worked very well in practice. It presents nontrivial, but surmountable, problems in agreement, coordination, and communication.

Instructors should keep the team sizes consistent, because smaller teams enjoy an advantage (they find it easier to reach consensus and earn the bonus points). But students in any size team will complain that teams of other sizes have an advantage.

5.4 Logistical Requirements

The activity requires an overhead projector with computer hookup. It works best in a room where students can work together in small groups (moveable chairs are crucial), preferably with access to blackboards or whiteboards.

The activity requires the following materials.

- For the instructor: slides for the lecture and game; scoring spreadsheet or PRS software (described in Section 5.5). All of these are provided with the Groupthink distribution (available from `mernst@csail.mit.edu`).
- For the students:
  - One copy of the handout per student, single-sided to permit easier reference.

- 16 small pieces of paper per student, for writing answers. This is necessary only if using manual vote collection technique (see Section 5.5).
- Optionally, token prizes for the highest-scoring team and the most improved team.

You may override the default team names in the spreadsheet (manual voting method) or program (electronic voting method).

5.5 Collecting Votes

There are two ways to collect votes and keep track of each team's score: manual and electronic. In the *manual* technique, each student writes his or her answer on a piece of paper, these are tallied by hand, and the result is entered into a provided spreadsheet. The spreadsheet tracks cumulative scores and (for round 2) improvement over the previous round. In the *electronic* technique, students use a hand-held remote control or other hardware to cast a vote, and the tallying and spreadsheet entry occur automatically. We provide software for the Personal Response System (PRS) from GTCO CalComp/InterWrite [7], which is a system of infrared remote control "clickers", but it could be ported to other hardware, including networks of workstations, PDAs, or cell phones.

Manual Voting Technique. For each question, students write their answer on a small piece of paper, turn it face down, and either give it to group's TA (or a designated team member) or place it in a common location such as the center of their table.

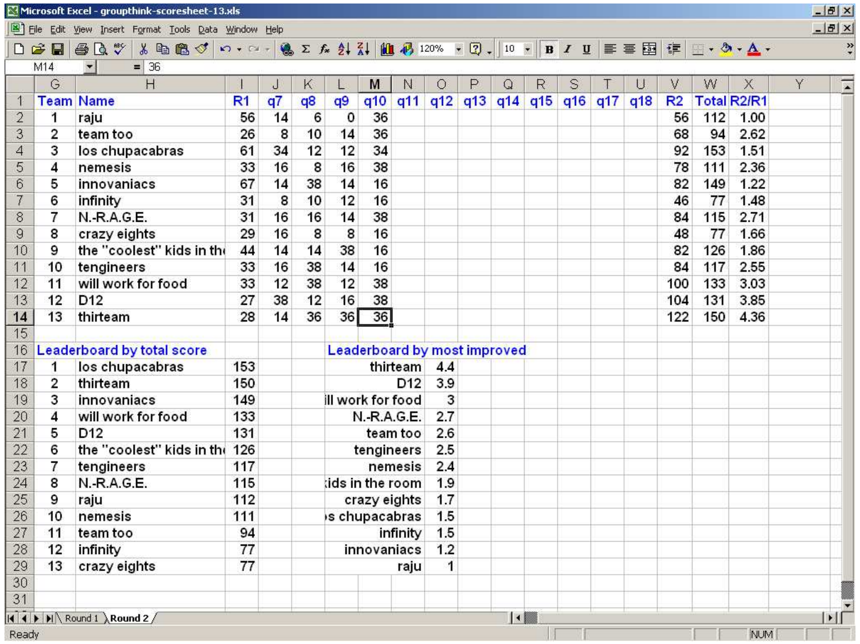
When all team members have voted, the TA reveals the answers. This ensures anonymity and prevents teams from blaming individuals who gave different answers than the plurality. Anonymity is important to make the participants think like a team: differences in answers are a failure of the team to reach consensus and to ensure understanding, not a failure of the individual. Having each student turn over his or her own answer would not achieve this goal.

It is necessary to know when each team is finished. We had up to 13 teams of 9 students, so we used the following mechanism (other mechanisms might work for smaller groups). The team's TA raises his or her hand as soon as the question is asked, then lowers it when all team members have answered. This makes it easy for the instructor to look around the room and see when everyone was done (when no more hands are up). One problem was that occasionally the designated person forgot to raise their hand and keep it raised until all answers were in. (The TAs preferred the use of the electronic voting technique described below.)

The instructor collects scores by calling out the name of each team, having that team's TA respond with the score, and typing these into a spreadsheet (see Figure 1). Communicating all the scores to the centralized location was somewhat clumsy, especially for large classes, and this was a complaint from some other institutions as well. (We mitigated the problem by collecting scores every two or three questions.) Projecting the spreadsheet so that everyone can see the current standings was popular with the students and increased the sense of excitement in the game.

Electronic Voting Technique. In order to overcome the clumsiness associated with manual vote tallying, we wrote custom software to support the Groupthink specification exercise. This software is distributed with the exercise for use by other instructors.

We had access to a classroom equipped with the InterWrite PRS, or Personal Response System [7]. This commercial system consists of remote control transmitters ("clickers") capable of transmitting a digit to infrared receivers that are positioned throughout the room. Typical uses for PRS are taking attendance and polling students during class to gauge comprehension. The vendor software has very limited functionality, so we wrote our own program instead, using the manufacturer's low-level interface to the hardware.



Team	Name	R1	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18	R2	Total	R2/R1
1	raju	56	14	6	0	36									56	112	1.00
2	team too	26	8	10	14	36									68	94	2.62
3	los chupacabras	61	34	12	12	34									92	153	1.51
4	nemesis	33	16	8	16	38									78	111	2.36
5	innovaniacs	67	14	38	14	16									82	149	1.22
6	infinity	31	8	10	12	16									46	77	1.48
7	N.-R.A.G.E.	31	16	16	14	38									84	115	2.71
8	crazy eights	29	16	8	8	16									48	77	1.66
9	the "coolest" kids in th	44	14	14	38	16									82	126	1.86
10	tengineers	33	16	38	14	16									84	117	2.55
11	will work for food	33	12	38	12	38									100	133	3.03
12	D12	27	38	12	16	38									104	131	3.85
13	thirteam	28	14	36	36	36									122	150	4.36

Leaderboard by total score			Leaderboard by most improved	
1	los chupacabras	153	thirteam	4.4
2	thirteam	150	D12	3.9
3	innovaniacs	149	ill work for food	3
4	will work for food	133	N.-R.A.G.E.	2.7
5	D12	131	team too	2.6
6	the "coolest" kids in th	126	tengineers	2.5
7	tengineers	117	nemesis	2.4
8	N.-R.A.G.E.	115	kids in the room	1.9
9	raju	112	crazy eights	1.7
10	nemesis	111	s chupacabras	1.5
11	team too	94	infinity	1.5
12	infinity	77	innovaniacs	1.2
13	crazy eights	77	raju	1

Fig. 1. The scoring spreadsheet for the manual scoring technique, during the second round of the game. The instructor types team scores into the top part of the spreadsheet. The two leaderboards (just one, during the first round) are updated automatically.

The software could be adapted to other systems that support in-class multiple choice questions [20]; academic systems include ClassInHand [19], the Digital Lecture Hall [17,13], and ActiveClass [14] (and its followons Classroom Presenter and Ubiquitous Presenter), and commercial systems include CPS [4], PRS [7], H-ITT [8], and TurningPoint [18].

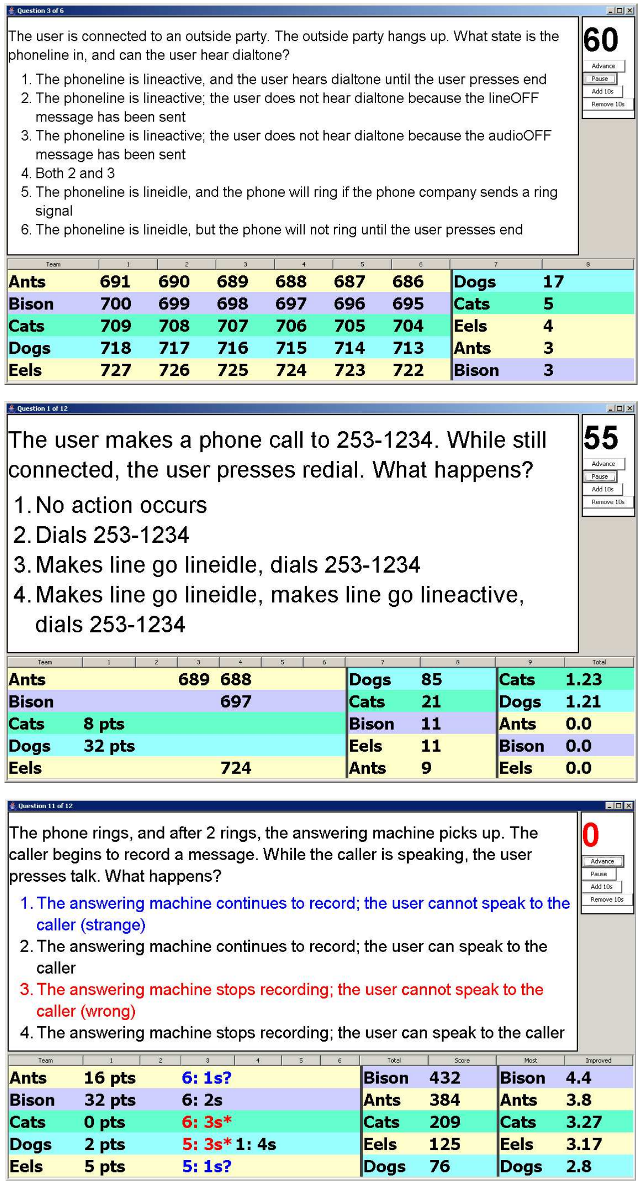


Fig. 2. User interface for the software for the electronic voting technique. The three screenshots were taken immediately after displaying a question, during voting on a question, and after voting was complete. The first screenshot was taking during the first round of the game, and the last two screenshots were taken during the second round of the game.

The program guides the class through the following phases:

Registration. Each student presses arbitrary buttons on his or her remote control and gets visible feedback. This makes students comfortable with use of the transmitter and gives them confidence that it is working properly. This also indicates which students are present, which is important for knowing when all members of a team have voted.

Questions. Figure 2 shows the user interface during voting. The top part of the UI displays a question, along with a countdown timer and a few controls for the instructor. The bottom part displays all transmitter IDs, along with the leaderboard that ranks the teams. (There are two leaderboards during the second round of the game, as indicated in the bottom two screenshots.) Color is used in the bottom part to help students find their team and their own transmitter ID.

As each person votes, the corresponding transmitter ID is erased in order to indicate that the vote has been received. The middle screenshot of Figure 2 shows the user interface after some of the participants have voted.

As soon as all members of a team have voted, the team's score is displayed (but not what the team members voted, as that could give guidance to other teams), and the leaderboard that ranks the teams is updated; see the middle screenshot of Figure 2.

As soon as all teams have voted, or time runs out, a summary of all votes for each team is displayed, as shown at the bottom of Figure 2. The notation "3: 1s 2: 3s" means that there were 3 votes that were "1" and 2 votes that were 3. To preserve anonymity (and prevent blame), votes are not reported by individual. Strange answers are displayed in blue (in both the top and bottom part of the user interface), and wrong answers are displayed in red.

Display winner. After the end of the first round, the winner of the first round is displayed. This heightens the subterfuge that the game is over.

Show changed rules The changed rules for round 2 are displayed (see Section 2.1).

Questions. The second round of questions is like the first, but with modified scoring rules (see Section 2.1) and with two leaderboards: one for overall score and one for most improved, as shown in the bottom two screenshots of Figure 2.

Display winners. The game ends with a display of two winners.

Use of the software greatly increased the pace of the game, especially for large classes, by eliminating pauses to type scores into the spreadsheet. The software was also quite popular with the TAs, who otherwise had to determine each team's score manually (and inevitably made mistakes). In fact, use of the software enabled us to give lower scores to the "strange" answers, which we feel is an improvement. We did have to change two of the questions to conform to the software. One question had had three parts and a total of 18 possible answers; we split it up into two separate questions with 6 and 3 possible answers, respectively. Another question had several multiple choice answers but also a write-in possibility; we eliminated the (rarely-used) write-in option. Use of solely free response questions would have been more challenging and might have had greater pedagogical value [15], but in our view the logistical problems, including subjective judgment of agreement, outweighed the benefits. Furthermore, Wolfman [20, p. 190] notes many other drawbacks of non-computer-based systems.

Acknowledgments

John Chapin of Vanu, Inc. co-conceived the Groupthink Specification Exercise and assisted in running it. Michael Gebauer wrote the PRS software for displaying questions and scores and collecting votes. Steve Wolfman, Rachel Pottinger, and Catherine Howell provided helpful feedback on this paper. The development of the Groupthink Specification Exercise was supported in part by the NSF and by Microsoft. This paper is an extended version of one that appeared in ICSE 2005 [5]. Finally, we thank the instructors who have used the activity and provided feedback. We welcome feedback from other instructors; send mail to `mernst@csail.mit.edu` to receive a copy of the Groupthink materials or to provide feedback or suggestions.

References

1. C. C. Bonwell and J. A. Eison. Active Learning: Creating excitement in the classroom. ASHE ERIC Higher Education Report 1, The George Washington University School of Education & Human Development, Washington, D.C., USA, 1991.
2. J. D. Bransford, A. L. Brown, and R. R. Cocking, editors. *How People Learn: Brain, Mind, Experience, and School*. National Academy Press, Washington, D.C., USA, expanded edition, 2000.
3. A. W. Chickering and Z. F. Gamson. Seven principles for good practice in undergraduate education. *AAHE Bulletin*, 39(7):3–7, Mar. 1987.
4. eInstruction. eInstruction — the global leader in interactive response systems. `http://www.einstruction.com/`.
5. M. D. Ernst and J. Chapin. The Groupthink specification exercise. In *ICSE'05, Proceedings of the 27th International Conference on Software Engineering*, pages 617–618, St. Louis, MO, USA, May 18–20, 2005.
6. R. M. Felder and L. K. Silverman. Learning and teaching styles in engineering education. *Engineering Education*, 78(7):674–681, 1988.
7. GTCO CalComp. InterWrite products. `http://www.gtcocalcomp.com/interwriteprs.htm`.
8. HyperInteractive Teaching Technology. Hitt (classroom remote system). `http://www.hitt.com/`.
9. C. Leiserson, B. Masi, C. Resto, and D. K. P. Yue. Development of engineering professional abilities in a co-curricular program for engineering sophomores. In *ASEE Annual Conference*, Salt Lake City, Utah, June 20–23, 2004.
10. J. J. McConnell. Active learning and its use in computer science. In *ITiCSE'96: the 1st Conference on Integrating Technology into Computer Science Education*, pages 52–54, Barcelona, Spain, June 1996.

11. W. J. McKeachie. *Teaching Tips: Strategies, Research, and Theory for College and University Teachers*. Houghton Mifflin, Boston, MA, USA, 10th edition, 1999.
12. J. A. Moon. Reflection in learning — some fundamentals of learning, part 1. In *Reflection in Learning and Professional Development, Theory and Practice*, chapter 9, pages 103–119. Kogan Page, Sterling, VA, USA, Jan. 2001.
13. M. Mühlhäuser and C. Trompler. Digital lecture halls keep teachers in the mood and learners in the loop. In *ELearn*, pages 714–721, Montreal, QC, Canada, Oct. 2002.
14. M. Ratto, R. B. Shapiro, T. M. Truong, and W. G. Griswold. The ActiveClass project: Experiments in encouraging classroom participation. In *CSCL'03: the International Conference on Computer Support for Collaborative Learning*, pages 477–486, Bergen, Norway, June 2003.
15. J. A. Siegel, K. J. Schmidt, and J. Cone. INTICE — Interactive Technology to Improve the Classroom Experience. In *2004 American Society for Engineering Education Annual Conference & Exposition*, Washington, D.C., USA, June 2004.
16. L. Springer, M. E. Stanne, and S. S. Donovan. Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis. *Review of Educational Research*, 69(1):21–51, Spring 1999.
17. C. Trompler, M. Mühlhäuser, and W. Wegner. Open client lecture interaction: An approach to wireless learners-in-the-loop. In *ICNEE'02: the 4th International Conference on New Educational Environments*, pages 24–46, Lugano, Switzerland, May 2002.
18. Turning Technologies, LLC. Audience response system, group response system. `http://www.turningtechnologies.com/`.
19. Wake Forest University Information Systems Research and Development Team. Classinhand. `http://classinhand.wfu.edu/`.
20. S. A. Wolfman. *Understanding and Promoting Interaction in the Classroom through ComputerMediated Communication in the Classroom Presenter System*. PhD thesis, University of Washington Department of Computer Science and Engineering, Seattle, Washington, 2004.

A Answerphone Module

The next three pages reproduce the student handout for the Answerphone module.

Groupthink Specification Exercise

In this exercise, you will design the control for a simple telephone with integrated answering machine. You will specify the telephone's behavior when the user interacts with it.

PART 1: Specifying Behavior

As a group, read this document and decide upon the behavior of the telephone under all possible user behaviors. Your design may be written down or agreed upon orally. We do not care how you record it.

PART 2: The Groupthink Game

After deciding on the behavior of the telephone, you will be given a variety of scenarios in which a user interacts with the telephone. Each member of your group will *individually* answer questions about the telephone's behavior. Your group is scored not on what your answers are, but whether all of the members' answers are consistent. However, your answers must satisfy the requirements and must be plausible behaviors that a user would find reasonable.

In a real project, consistent answers would lead to components that interoperate correctly, behavior that is consistent with the documentation, etc. Problems due to diverging interpretations are common in software (and other!) development teams where the specification is ambiguous or underconstrained. We encourage you to think hard in part 1!

Here is an example question:

> The user is connected to an outside party. The outside party hangs up. What state is the phoneline in?
>
> A. Lineactive (the user hears dialtone)
>
> B. Lineidle (the user does not hear dialtone)

The group that wins the Groupthink Game will receive a prize. Your group may not give answers based on the form of the game; for instance, you may not agree to answer "A" if you aren't sure what else to do.

Definitions

lineidle — The phone is hung up or "on hook." In a traditional phone, this means the handset is lying in the cradle, but your phone uses the **`end`** key instead.

lineactive — The phone is picked up or "off hook." In a traditional phone, this means the handset is not in the cradle (it is "off hook"), but your phone uses the **`talk`** key instead.

ring signal — A +/- 24 volt AC signal sent over the phone line, which causes a traditional phone to ring. The phone company only sends a ring signal if it detects the lineidle state.

System Specification

TELEPHONE COMPONENTS

- Handset (includes both speaker and microphone)
- 24-character display
- Answering machine
- Keypad with keys labeled **`talk`**, **`redial`**, **`ansmachine`**, and **`end`**.

  Simplification: The keypad also has 0 through 9, but in this exercise, you can ignore how those keypresses are handled. When the **`talk`** *key is pressed, the digits previously entered by the user are delivered to the control software (much like a cellular phone). The* **`redial`** *key does not deliver any numbers. There is no hook or cradle as with a traditional phone, just the keys.*

FUNCTIONS

- The user places a call by pressing **`talk`** or **`redial`**. The user answers a call by pressing **`talk`**.

  Simplification: Your phone is not required to handle call waiting.
- The user begins using the answering machine by pressing **`ansmachine`** on the handset.

  Simplification: In this exercise, you will not be asked to specify the answering machine's behavior during message review.
- The user presses **`end`** to end a call or to stop using the answering machine.

REQUIREMENTS

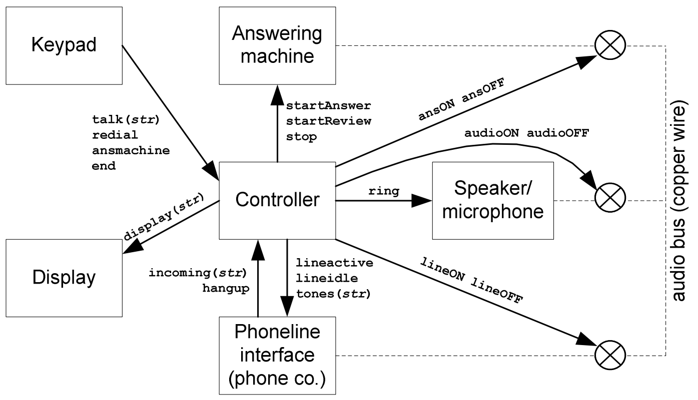
- The display must show the appropriate information at all times.
  - If idle — show "READY"
  - If a ring signal is being sent by the phone company — show the caller ID information of the caller
  - If connected to an incoming call — show the caller ID information of the caller
  - If connected to an outgoing call — show the number being called
  - If using the answering machine — show "ANSWERING MACHINE"
- If a ring signal is delivered, the telephone must ring and show the caller ID of the caller. If the user doesn't answer the call within 2 rings, the answering machine must pick it up.

CHALLENGE

This specification may be incomplete or inconsistent. This is normal in any development effort! Your group should figure out the details needed to handle all possible scenarios that you might be asked about in the Groupthink Game.

System Architecture

The telephone has the following components. The messages that may be exchanged between the handset controller and the other components are labeled in the diagram. Analog audio links are shown with dashed lines. Switches (represented by $\otimes$) either make or break audio connections.



`talk(`*`string`*`)`	The user typed the digits in the argument string and then pressed **`talk`**
`redial`	The user pressed **`redial`**
`ansmachine`	The user pressed **`ansmachine`**
`end`	The user pressed **`end`**
`display(`*`string`*`)`	Makes the LCD display show the characters in ***`string`***, a 24-character string
`startAnswer`	Play outgoing message and record the caller's message
`startReview`	Play back recorded messages and perform other user interactions
`stop`	Stop answering machine functions, return to idle state
`incoming(`*`string`*`)`	The phone company sent a ring signal with ***`string`*** as caller ID information. This message is repeatedly sent (every 6 seconds) until the call is answered or the caller hangs up.
`hangup`	The phone company indicates that the remote party has hung up
`lineactive`	Put the resistance across the phone line that indicates the phone is active
`lineidle`	Put the resistance across the phone line that indicates the phone is idle
`tones(`*`string`*`)`	Send the digits in ***`string`*** out over the phoneline as touch-tones
`ring`	Causes the speaker to play one ring tone
`ansON ansOFF`	Connect/disconnect the answering machine to the audio bus
`audioON audioOFF`	Connect/disconnect the speaker and microphone to the audio bus
`lineON lineOFF`	Connect/disconnect the phoneline to the audio bus

Simplification: Messages among telephone components are never lost or corrupted.