FEATURE

Kevin W. Bowyer

"Star Wars" Revisited

Ethics and Safety-Critical Software

Safety-critical software is a core topic in courses on "ethics and computing" and "computers and society." It is also a core topic in software engineering courses. In the 1980s, the U.S. Reagan-era Strategic Defense Initiative was the focus of a great deal of technical argument relating to design and testing of safety-critical software. Today, most students in the U.S. have no familiarity with the substance of these arguments. However, with U.S. presidents Clinton and Bush considering various versions of a national missile defense system, the topic has again become relevant and applicable to current events.

Fig. 1. U.S. President Ronald Reagan's 1983 speech is the source of high-level requirements for the "Star Wars" system.

Work on this module began under support of National Science Foundation Undergraduate Faculty Enhancement grant DUE-9752792 [16]. The video of Reagan's SDI speech was obtained with the help of the Reagan presidential library. Video clips from the CPSR-sponsored M.I.T. debate are used with permission of the national office of the Computing Professionals for Social Responsibility. Portions of this paper were presented at the 2001 Frontiers in Education conference and the 2001 International Symposium on Technology and Society conference [20].

The author is Schubmehl-Prein Department Chair in the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556. Email: kwb@cse.nd.edu.

0278-0079/02/$10.00©2002IEEE

The topic of a national missile defense system has again become relevant.

This article describes a curriculum module developed around a Reagan-era SDI debate on the theme – “Star wars: can the computing requirements be met?” This module may be appropriate for use in ethics-related or software-engineering-related courses taught in undergraduate Information Systems, Information Technology, Computer Science, or Computer Engineering programs. It should also be appropriate for use in courses in general engineering ethics or technology and society.

The Reagan-era “Star Wars” Defense Program

The Reagan-era “Star Wars” ballistic missile defense program generated a great deal of controversy. One aspect of this controversy involved the design and testing of safety-critical software. In 1985, the Computing Professionals for Social Responsibility (CPSR) sponsored a debate, held at M.I.T., on the question – Star Wars: Can the Computing Requirements Be Met? Controversy on this particular point was sparked by, among other things, David Parnas’ resignation from the Strategic Defense Initiative (SDI) computing panel. Parnas argued that it was impossible, in principle, to create SDI software that would allow a useful level of trust in the system. He presented his argument for this conclusion at the CPSR-M.I.T. debate and in various publications. Chuck Seitz, a member of the SDI computing panel who did not resign, argued at the debate in favor of the feasibility of SDI software. Michael Dertouzos served as debate moderator. Joseph Weizenbaum, who was not a member of the SDI panel, argued the con position along with Parnas. Danny Cohen, who served as chair of the SDI panel, argued the pro position with Seitz.

Fig. 2. Michael Dertouzos gives an SDI System Overview at the M.I.T.-CPSR Debate.

The presentations at this debate, in particular those of Parnas and Seitz, provide the core for developing a curriculum module that deals with ethical issues involved in the creation of safety-critical software. The module should be appropriate for use in courses on software engineering, ethical issues, social impact of computing, or technology and society. It has been successfully used both in courses aimed at first-year students who are not yet (and may not become) computing majors, and in a senior-level “capstone” course for Computer Science and Engineering majors. Some level of programming experience will of course help students to appreciate the complexities of software testing and debugging. Some level of discrete math background should help students to appreciate reliability-related concepts such as statistical independence of failures, but is certainly not necessary in order to understand the essence of the larger argument.

This course module can be viewed as divided into five sections:

1) introduction to the basic SDI problem,

2) evaluation of Parnas’ argument that trustworthy SDI software is not possible,

3) evaluation of Seitz’ argument that trustworthy SDI software is possible,

4) connection to current ballistic missile defense efforts, and

5) consideration of ethical issues for computing professionals working on such projects.

The first section of the module should give students a basic understanding of the requirements of an SDI system, and make it clear that this is an extreme instance of safety-critical software. The second and third sections present the arguments against and for the feasibility of creating trustworthy software for an SDI system. These sections contain the major technical substance of the module from a computing perspective. The purpose of the section on connecting the Reagan-era arguments to current missile defense efforts is to assess the modern relevance of conclusions in the original argument. The purpose of the last section of the course is to explicitly consider important ethical issues involved in this case study.

The content of each section of the module is outlined in more detail below.

Understanding the Context of the SDI Problem

The section of the module on understanding the SDI problem incorporates a short video clip from President Ronald Reagan's "Star Wars speech" (see Fig. 1) delivered in March of 1983 [1], and a clip from the 1985 CPSR-M.I.T. debate (see Fig. 2) in which moderator Michael Dertouzos gives an overview of the SDI scenario and requirements. Dertouzos outlines parameters of the SDI scenario, such as the size of the geographic area to be monitored for an attack launch, the projected time span of an attack, and the number of missiles, warheads and decoys that might be involved.

The goal of this section of the course is for students to work through a general understanding of the issues in the systems analysis and requirements specification stages of SDI software development. The PowerPoint material makes references to the waterfall model of software develoment, not to endorse this model over other models, but to focus students' thinking on the problems inherent in specifying requirements for such software.

It is important that students develop an appreciation for the extreme difficulty of the SDI computing problem. For instance, at one point Dertouzos mentions that planners envision that the SDI system will maintain "a consistent distributed database" of the missile tracking information. There is some audible laughter from the audience at this point, because the demands of "consistent" and "distributed" are inherently contradictory at some level. This point may not be readily apparent to students as they watch the video. Therefore it may be useful to explicitly point out the difficulty involved in the real-time nature of the problem, the distributed communications and control, and the automated interpretation of sensory data that may vary with the state of nature and the intentions of an intelligent adversary.

Then-current thinking about the SDI scenario and technology is well represented in the "Eastport Report" and an U.S. Office of Technology Assessment report [22],[23]. Electronic copies of these government documents are available on CD with the video clips and PowerPoint for this curriculum module.

UNDERSTANDING PARNAS' ARGUMENT

The purpose of this module of the course module is for the students to work through a summary of Parnas' technical argument for why it is not possible to create trustworthy SDI software. This section of the module incorporates a video clip of Parnas' presentation (Fig. 3) and additional PowerPoint slides. Any of several papers by Parnas might be used as references or handouts with this section (e.g., [2].) The PowerPoint material includes slides that ask students to identify the conclusion advanced by Parnas, and then, given the conclusion, to identify the premises used to argue for this conclusion. Students should also develop a clear idea of Parnas' reasons why the SDI computing problem is more difficult than other complex computer systems. For example, launch of a space shuttle can be delayed if computer and weather conditions are not satisfactory, control of a nuclear power plant does not require defeating the intentions of an intelligent adversary, and other sophisticated weapons systems are used many times and so can be debugged after initial failures.

The Reagan-era "Star Wars" ballistic missile defense program generated a great deal of controversy.

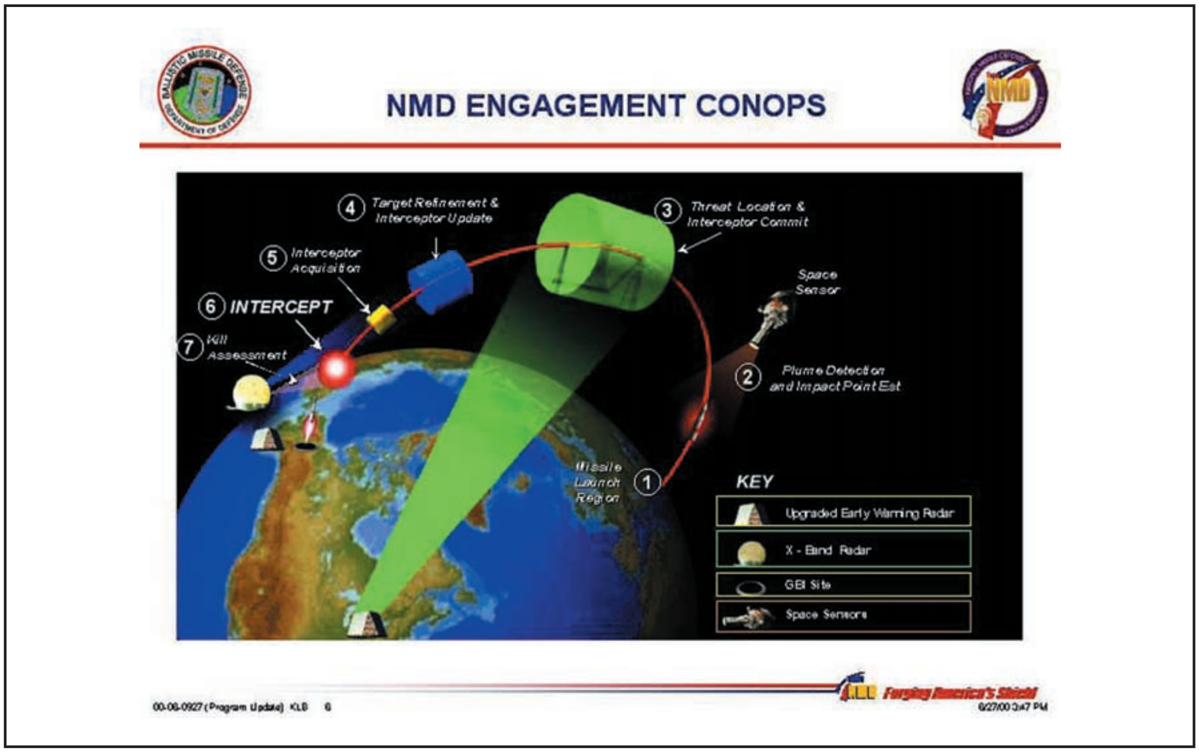
Fig. 3. Parnas presents an argument that trustworthy SDI software is not possible in principle.

Students may need some guidance in formalizing the structure of Parnas' argument. His presentation does contain a clear technical argument in reponse to the topic defined for the debate – Star Wars: can the computing requirements be met? However, he also goes beyond this at times and suggests conclusions of larger socio-political questions. Students may be tempted to assert that he argues for conclusions such as "The United States should not pursue SDI" or "Pursuing SDI will make the United States weaker rather than stronger." In fact he does, but students should be able to realize that these are not conclusions of the immediate computer systems engineering argument. Students should be encouraged to focus primarily on the argument that relates to the technical issue of whether it is possible, in principle, to create SDI software that could be considered trustworthy. It is possible that some students will have passionately-held opinions about peace, strong defense, or President Reagan's legacy. Again, these are probably not appropriate as the immediate focus of class discussion.

Students find it easier to reach an appropriate summary of Parnas' argument if they are first guided to a statement of the conclusion. Discussion of different possible conclusion statements and how they relate to the debate topic should bring students to a statement similar to – "It is not possible to construct SDI software that could confidently be expected to work correctly when needed."

Fig. 4. Seitz presents an argument for the feasibility of creating an SDI System.

The "confidently" qualifier is a potential source of ambiguity. However, Parnas suggests that a pragmatic definition is the level of confidence that you have that your car will start when you turn the ignition key. This may provide an opportunity for useful class discussion about what constitutes an appropriate level of confidence and whether or how such confidence might be measured. Other analogies can be offered similar to that of the car starting: for instance, the confidence you have that your computer system will correctly retrieve a file from disk when it is requested. Most examples that students propose in class will likely not incorporate the complication of an intelligent enemy. This point might be made by suggesting a sports-related analogy. For example, what is your level of confidence that the opposing team will not be able to score given that your team correctly executes the defense it has planned ahead of time? The point that Parnas makes is that our confidence that the software will work correctly when needed is directly linked to our assumptions about how an intelligent adversary will choose to structure an attack.


NMD ENGAGEMENT CONOPS

4 Target Refinement & Interceptor Update
3 Threat Location & Interceptor Commit
5 Interceptor Acquisition
6 INTERCEPT
7 Kill Assessment
Space Sensor
2 Plume Detection and Impact Point Est
Missile Launch Region 1
KEY
Upgraded Early Warning Radar
X - Band Radar
GBI Site
Space Sensors
Forging America's Shield

00-06-0927 (Program Update) KLB 6

6/27/00 3:47 PM

Fig. 5. Depiction of missile defense scenario from http://www.acq. osd/mil/bmdo/.

TABLE I
CATEGORIZATION OF COMPLEX SYSTEMS ACCORDING TO APPLICATION CONSTRAINTS

	Property of the application that complicates design and testing					
Computer System Application	real-time response requirements	"signal-to-symbol" sensor data processing	uncontrolled sensor imaging conditions	intelligent adversary motivated to fool system	starting conditions controlled by adversary	requires coordinated distributed computing
Chess-playing	No	No	No	Yes	No	No
Telephone switching	Yes	No	No	No	No	Partially
Space shuttle	Yes	Yes	Yes	No	No	No
Nuclear power plant	Yes	Yes	Partially	No	No	No
Fighter jet	Yes	Yes	Yes	Yes	Partially	No
SDI	Yes	Yes	Yes	Yes	Yes	Yes

Once students have the conclusion of the argument, they should be able to identify relevant premises that Parnas uses to argue for the conclusion. Important elements of the technical arugment have to do with the specifications being unkown, there being no practical way to realistically test the software, and there being no time to debug the software in use. While factors such as the number of programmers required to work on the software and the estimated size of the system may also be relevant, Parnas explicitly asserts that his argument is independent of the size of the software.

As a result of analyzing the material in this section, students should be able to reach a summary of Parnas' technical argument similar to the following. It would also be within the spirit of Parnas' presentation to give a one-premise form of the argument. The statement – "Since the specifications are inherently unkown, therefore it is not possible to know whether you have written the desired system." – would reasonably capture the essence of the argument.

Candidate Summary of Parnas' Argument –

Since:

(1) The specifications for the software cannot be known with any confidence, because they depend on the actions of an intelligent adversary, and

(2) The software cannot undergo any fully realistic testing, because this would require realistic sensor data reflecting the (unkown) scenario for enemy attack, and

(3) There would be no time during an attack to repair and reinstall failing software ("no real-time debugging"),

Therefore: It is not possible to construct SDI software that could be confidently expected to work correctly the first time it is needed.

Parnas mentions a number of items during his presentation that should be defined for the class in order for them to get the most out of his presentation. Among these are a) the acronym MAD, standing for Mutual Assured Destruction, the cold-war strategy that says nuclear war is best deterred by having each side believe that it would result in mutual destruction, b) ADA, the programming language, in the context of it being an ambitious software project that took a number of years to result in reasonably efficient and correct compilers, c) "people with Dutch accents," indicating Edsgar Dijkstra, in the context of suggestions that the problem with software is that the software engineers are not talented enough, d) "Byzantine agreement," a formalism of the problem in which *N* distributed systems communicate to reach agreement among the correctly-working systems even when some fraction of the *N* systems may send false messages, e) "Safeguard," referring to an early ballistic missile defense system intended to defend only selected sites neces-

Parnas argued that it was impossible, in principle, to create SDI software that would allow a useful level of trust in the system.

sary for the U.S. to launch a retaliatory strike, f) "Vietnam," in the context of the weapons systems used in that war, g) "someone named Walker," meaning a person with Defense Department security clearance who is discovered to be a long-time spy for the enemy, and h) a reference to Fred Brooks, in the context of a person of distinguished reputation in software engineering [21].

Parnas also makes an argument that the SDI computing requirements are, in effect, unique and more difficult than those for any other complex system that might be selected for an analogy. At one point in the debate, Cohen mentions the space shuttle as an example of a system requiring large and complex software. Parnas' response is that whereas NASA can delay a launch up until the last second, the president cannot call up the (former) United Soviet Socialist Republic (USSR) to delay a nuclear war. An interesting class exercise would be to make a list of constraints on the SDI computing system and ask students to classify other complex systems according to these contstraints in order to find a good analogy. The result might be something like that in Table I.

Students who have a strong a priori belief in the positive value of ballistic missile defense may feel that their political beliefs are being challenged.

Understanding Seitz' Argument

Similar to the section on Parnas' argument, the point of this section is for the students to work out a critical-thinking summary of Seitz' argument. Also similar to the previous section, the material for this section includes video of Seitz' presentation (see Fig. 4), plus supporting PowerPoint slides. The premises should represent technical bases that Seitz uses to argue for his conclusion. Seitz quotes from the SDIO computing panel (from which Parnas resigned and Seitz and Cohen did not) as part of his presentation.

One quote is: "The feasibility of the battle management software and our ability to test, simulate, and modify the system are very sensitive to the choice of system architecture. In particular, the feasibility of the battle management software is much more sensitive to the system architecture than it is to the choice of software engineering techniques." From this it seems clear that Seitz is arguing that the problems can be solved through an appropriate choice of system architecture.

As with Parnas' presentation, students may need some guidance to arrive at an appropriate summary of Seitz' argument. Seitz uses relatively few terms in his presentation that will require definition for the class. One concept that students may not be familiar with is the "signal to symbol" transition in processing sensor data. This refers to the process of moving from raw sensor data to a symbolic description of entities in the data. The raw data might be a 2-D array of non-negative integers that form an image representing some property such as heat, refelected light, or distance from the sensor. The symbolic description might be something like "missile centered at location *x,y*".

The conclusion of Seitz' argument should be a statement to the effect that it is possible to construct reliable SDI software. The premises of the argument will have to do with hierarchical control structures being well understood, conceptual control structure that coincides with physical control structure being an advantage for reliable implementation, and modularity being an advantage in implementation and testing. It should be possible for students to arrive at a summary of Seitz' argument similar to the following:

Candidate Summary of Seitz' Argument –

Since:

(1) Hierarchical control structure is natural and well understood, and

(2) Hierarchical organization seems attractive both for the conceptual flow of data abstraction, and the physical organization of the system, and

(3) Hierarchical organization naturally leads to modularity, which is an advantage for achieving reliable implementation,

Therefore: It is possible to construct SDI software that could be confidently expected to work correctly the first time it is needed.

With the arguments of the two sides of the debate identified, students should begin to have the basis for developing their own informed opinion on the issue. Students can also be asked to assess stylistic issues in the presentations, and how these factors might influence the effect on a non-computing-literate audience.

For example, how does the use of personal comment and sarcasm affect the communication of technical content? How does/would an explicit premise-conclusion summary of the argument aid the audience's understanding? And how does not responding explicitly to an opponent's asserted premises affect credibility?

At the end of analyzing the two presentations, it will be clear to most students that Parnas' technical argument is essentially correct and is not refuted by Seitz' argument. One over-simplified characterization of the debate is that Parnas says "We can't test it" and Seitz then replies "We can build it." In this sense, the two presentations do not respond equally well to the theme of the debate, "Star Wars: can the computing requirements be met?" Seitz argues that we can build something that should be useful, but does not really address the issue of how to test that it would meet requirements. Parnas argues that it doesn't make any difference what is built or how it is built, because there won't be any means of testing that it meets requirements.

For many people, the in-principle nature of the point about the specifications for the software being unknown is enough to carry the argument by itself. The rather clear-cut nature of this narrow technical argument is a potential pitfall for use of this material. Students who have a strong *a priori* belief in the positive value of ballistic missile defense may feel that their political beliefs are being challenged, or that the material has somehow been unfairly presented. There are several points to consider in this regard. One point is that the purpose of the study is, in so far as possible, to discover the truth, and this may result in a challenge to *a priori* beliefs. A second point is that the two presenters, Parnas and Seitz, are both accomplished, word-class computer scientists. This point can, and probably should, be emphasized through a pre-class assignment described later.

It also should be pointed out that both Parnas and Seitz came to the debate already familiar with the essence of the other person's argument. Parnas was presenting arguments that he had already published and that Seitz certainly would have known about. Similarly, Seitz was presenting arguments based on the published report of the SDI computing committee. It is not reasonable to think that either person was caught unaware by the other person's argument.

A point that might merit exploration in courses on technology and society is that the narrow question of whether or not it is possible to create trustworthy SDI software does not necessarily answer the general question of whether or not it is worthwhile to attempt to construct an SDI system. Students may find this point a bit paradoxical. However, one defense sometimes raised by supporters of the Reagan legacy is that the Soviet attempt to respond to the SDI program was an important contributing factor in the breakdown of the Soviet Union [26]. That is, even if SDI did not or could not work, it aided a larger objective of "defeating" the Soviet Union. Similar sorts of arguments were made in 2001 by President George Bush's Secretary of Defense Donald Rumsfeld. He expressed "that the United States is likely to deploy certain antiballistic missile systems before testing on them is completed" [24] and argued that "even if a missile defense system does not work properly, it would make an adversary think twice before launching a missile at the United States" [25]. Thus the Bush administration appears willing to stipulate that the system may not work properly or well, but is willing to undertake the expense of building a system anyway in order to achieve other perceived benefits. This clearly illustrates how there may be a distinction between the technical question and related political questions.

Relation to Current Missile Defense Scenarios

The point of this section of the course is to relate the evaluation of Parnas' and Seitz' arguments to current ballistic missile defense plans. A recent special issue of *IEEE Spectrum* assesses the state of various U.S. missile defense programs [4]. Overall, the web site of the DoD Ballistic Missile Defense Office (BMDO) is an excellent source of information [6]. The is perhaps especially true because the envisioned scenarios for a missile defense system are evolving over time. Information from this site should be useful to summarize the current official scenarios, plans, and status. An example figure from this web site appears in Fig. 5.

The U.S. continues to spend large amounts of money on missile defense. An editorial in *Science* magazine in 2001 estimated cumulative U.S. expenditures on missile defense at $100 billion, in current dollars [7].

Missile defense is of course a socially and politically controversial topic. Numerous articles on missile defense are also available in the popular press (e.g., [9] "con" and [12] "pro"), and numerous interest groups have web pages with archives of press releases and news reports on the topic.

Reviewing the recent history of U.S. ballistic missile defense efforts can give valuable perspective on the feasibility of the goals of the Reagan-era program [3] – "In the last 15 years, the United States has conducted 20 hit-to-kill intercepts, for the BMD programs discussed here as well as in other tests. Six intercepts were successful; 13 of those intercepts were done within the last five years, and among them, three intercepts succeeded. ... no real attempts have been made to intercept uncooperative targets — those that make use of clutter, decoys, maneuver, antisimulation, and other counter measures. Nor have any tests attempted to use a real battle management system that integrates data from a diverse array of actual tracking sensors and directs an interceptor to a target." An interesting assignment for students may be to gather information on the most recent tests and to assess the level of realism in the tests (clutter, decoys, ...).

Students with any previous software engineering course work should easily realize that the testing done to date does not begin to address the more difficult technical issues indentified in the Reagan-era debate. Tests that use data from actual tracking sensors and that try to hit targets that employ simple counter-measures would be only the beginning of "realistic" testing. Increased realism would include, for example, multiple targets that create various loads and structures of attack, varied weather conditions, and simulation of random and coordinated failures in the system due to attack. A useful exercise for students may be to ask them to sketch a plan for several levels of increasingly realistic testing of SDI software. Real incidents from actual tests of missile defense technology can be used to emphasize the difficulty involved. For example, in a 1997 test "the clouds had cleared but a software problem caused the laser to recycle, or unexpectedly lose power, during the brief period in which the satellite was within range" [10].

It is important for students to realize that the motivating scenario for current missile defense efforts is not the same as for the Reagan-era program. Potentially important differences include 1) the anticipated size and sophistication of an attack, and 2) the geographical location/size of the hypothesized enemy. The Reagan-era SDI program envisioned an attack of tens of thousands of missiles coming from the area of the former USSR. Current thinking envisions an "attack by a rogue state using a handful of warheads outfitted with relatively simple countermeasures" or "an accidental launch of a few warheads by Russia or China" [3]. This clearly reduces, to some degree, the required complexity of the ballistic missile defense system. How this affects the conclusions of the Parnas-Seitz debate is not entirely clear, and provides an opportunity to pursue an interesting line of reasoning. When asked, most people will feel that a successful SDI system for the currently envisioned scenarios is perhaps possible, or at least is not as clearly impossible as for the Reagan-era scenario.

This feeling presents something of a conflict, because Parnas explicitly made an "in principle" argument. When students accept Parnas' argument for the Reagan-era scenario, but feel that it might be possible to construct a reliable system for current scenarios, there is a need to resolve the apparent inconsistency. The resolution appears to lie in the perceived feasibility of "over-engineering" the system. By "over-engineering" we mean designing a system explicitly to have substantial over-capacity relative to the size of the threat, akin to the old engineering idea of a "margin of safety" in the design. With the Reagan-era scenario of tens of thousands of warheads and hundreds of thousands of sophisticated decoys, most people could not imagine over-engineering the system to a degree that would provide confidence. With current more limited scenarios, it seems easier for people to imagine that the system might be built with enough excess capacity to provide confidence that it would work in the presence of some failures. Whether or not this is truly a realistic option will of course depend on the particular assumptions made about the size and sophistication of the threat.

Given that the majority of the class accepts the in-principle argument made by Parnas, but then also believes that a missile defense system for current scenarios is feasible, it makes sense to explore the differences between the scenarios. Students might be asked to rate the feasibility of constructing a missile defense system in various in-between scenarios. For example, what if the "rogue country" in current scenarios could launch hundreds, thousands, or tens of thousands of missiles? Or, what if the enemy was able to launch the attack from unknown points over a larger geographic area? The point of the exercise would be to isolate the factors of the scenario that appear to most affect feasibility of the system.

A current whistle-blowing case alleges fraud in the testing and development of software in recent missile defense efforts [19]. The whistle-blower, Nira Schwartz, alleges that TRW knew that the performance of its software to discriminate warheads from decoys was far below what was reported to the government. The allegations have been investigated at several levels. One Pentagon criminal investigator said that there is "absolute irrefutable scientific proof that TRW's discrimination technology does not, cannot, and will not work" and that TRW was "knowingly covering up its failure" [19]. A team put together to look at the allegations and the report said that the TRW computer programs "were 'well designed and work properly' provided that the Pentagon does not have wrong information about what kinds of warheads and decoys an enemy is using" [19]. In other words, if one assumes specifications for the warheads and targets that an enemy will use, and if this information turns out to be correct, then the software should work. This rather clearly shows that one critical weakness is unknown specifications – the same weakness that Parnas emphasized over fifteen years earlier!

The coverage of this whistleblowing incident also provides excellent opportunities for critical-thinking exercises. Congressman Curt Weldon of Pennsylvania provides several quotes arguing for the construction of a missile defense system. One of these is as follows: "If we don't build a new aircraft carrier, we have older ones. If we don't build a new fighter plane, we have older ones. If we don't build a new tank, we have older ones. If we don't build missile defense, we have nothing" [19]. In premise-conclusion form, his argument appears to be:

Since:
We have existing but older forms of many weapons systems, and
We have no existing form of a missile defense system,
Therefore: we should build a missile defense system.

As is often the case, the argument loses some of its appeal simply by being cast into explicit premise-conclusion form. The argument does not address cost tradeoff issues such as whether it would be better to have a missile defense system or newer versions of other weapons systems (or other security-enhancing measures). More fundamentally, it also does not address the issue of whether it is even possible to construct a reliable missile defense system. For students that would understand the halting problem, the following might be offered for discussion as a possibly analogous argument:

Since:
We have existing but older forms of many software development tools, and
We have no existing tool to check for whether a program will run into an infinite loop,
Therefore: we should build a tool that will check whether a program will run into an infinite loop.

The goal here sounds great, but there is computer science theory that says it is impossible. Some software engineers might regard the idea of constructing software to meet unknown specifications as similarly impossible. The pragmatic response is that some specifications will be assumed that will hopefully cover the real-world cases that arise.

Congressman Weldon also makes an analogy between critics of the President Kennedy's program to land a person on the moon and current-day critics of missile defense [19]. The intent of the argument is apparently to have people conclude that the SDI program would succeed in the way that the program to land a man on the moon succeeded. The big missing element in this analogy should be clear. The moon-landing program had to deal with problems presented by nature, whereas the missile defense program has to deal with problems presented by an intelligent enemy that is motivated to defeat the system. Again, this point relates back to the issue of unknown specifications.

Another quote from Congressman Weldon came in response to a letter signed by Nobel Laureates arguing against development of missile defense. Weldon's comment was: "Well, I don't know any of them that's come to Congress or to me. I've not seen one of their faces. I mean, you know, it's easy to get anyone to sign a letter. I sign letters all the time" [19]. In premise-conclusion form, the argument appears to be:

Since:
I have not talked to them face-to-face, and
They have only written a letter, and I sign letters all the time (it is easy to get me to sign a letter),
Therefore: their letter does not mean anything.

There is one particularly telling point here. It seems that the congressman's argument uses a premise that the Nobel laureates' letter should not be taken seriously because, by analogy, he signs letters all the time that he does not mean to be taken seriously. In any case, again, the response does not address any of the issues of substance. The analysis of these quotes may be more relevant to classes in technology and society than to classes in software engineering, and should serve to emphasize to students that the political decision-making about missile defense is taking place in a notable absence of any serious technical discussion.

After covering the material in this section of the module, students should understand how the current national missile defense scenarios relate to the technical arguments developed during the Reagan-era SDI debate. They should also appreciate the fact the much of the current political discussion about national missile defense is seriously lacking in consideration of technical feasibility.

Relation to Codes of Ethics

Discussion of this case study should include consideration of ethical issues that confront computing professionals working on such projects, with explicit reference to the different professional codes of ethics. Students should be asked to evaluate the ethical issues relative to the professional codes of ethics, and project what they might do in various situations. Among the many questions that students might be asked to address are:

1) Was Parnas right in resigning his $1000/day consulting position to "blow the whistle" on the SDI program?

2) Is it ethical today to accept work on national ballistic missile defense systems, or, more generally, on systems that you believe cannot possibly work as advertised?

3) Assume that you believe it is ethical to work on national ballistic missile defense systems, and that you are a manager at a company doing such work – how should you treat an employee who believes that it is ethically wrong to work on such systems?

4) How should you, as a professional, respond to a non-computing-literate person who asks you if a national ballistic missile defense system is possible?

The various codes of ethics for the computing professions offer some fairly clear guidance on such questions. Relevant items of the Association of Information Technology Professionals' (AITP) standards of conduct [5], [15] that students should consider include the following:

> "In recognition of my obligation to society I shall: ... Use my skill and knowledge to inform the public in all areas of my expertise. ... To the best of my ability, insure that the products of my work are used in a socially responsible way. ... Never misrepresent or withhold information that is germane to a problem or situation of public concern nor will I allow any such known information to remain unchallenged.
>
> In recognition of my obligation to my fellow members and the profession I shall: ... Cooperate with others in achieving understanding and in identifying problems."

Relevant elements of the Association for Computing Machinery (ACM) code of ethics include (numbers identify specific sections and items of the full code [15]):

> "As an ACM computing professional I will... [2.5] Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks. … [2.7] Improve public understanding of computing and its consequences.
>
> "As an ACM member and an organizational leader, I will... [3.4] Ensure that users and those who will be affected by a computing system have their needs clearly articulated during the assessment and design of requirements. Later the system must be validated to meet requirements."

Relevant elements of the ACM/IEEE-Computer Society (CS) Software Engineering Code of Ethics include the following (numbers identify specific sections and items of the full code [15], [17]):

> "Software engineers shall act consistently with the public interest. In particular, software engineers shall, as appropriate:... [1.3] Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good. ... [1.4] Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.
>
> "Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest. In particular, software engineers shall, as appropriate: ... [2.6] Identify, document, collect evidence and report to the client or the employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate intellectual property law, or otherwise to be problematic.
>
> "Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. In particular, software engineers shall, as appropriate: ... [3.2] Ensure proper and achiev-

TABLE II
Topic and Length of Video Clips Used in the Presentation

President Reagan's call for SDI program	0:42	Charles Seitz' argument for feasibility	16:38
Michael Dertouzos' overview of SDI	6:01	David Parnas' rebuttal	5:35
David Parnas' argument against feasibility	22:13	Charles Seitz' rebuttal	2:31

> able goals and objectives for any project on which they work or propose. ... [3.7] Strive to fully understand the specifications for software on which they work. ... [3.8] Ensure that specifications for software on which they work have been well documented, satisfy the users requirements and have the appropriate approvals. ... [3.10] Ensure adequate testing, debugging, and review of software and related documents on which they work.
>
> "Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance. In particular, those managing or leading software engineers shall, as appropriate: ... [5.12] Not punish anyone for expressing ethical concerns about a project."
>
> "Software engineers shall be fair to and supportive of their colleagues. In particular, software engineers shall, as appropriate: ... [7.5] Give a fair hearing to the opinions, concerns, or complaints of a colleague."

Students should be encouraged to consider how they would hope to respond to the ethical issues when/if they face them in their career, and to evaluate their anticipated responses in the context of the codes of ethics. Answers will not be easy for some questions. For example, the ACM/IEEE-CS Code of Ethics requires software engineers to "act consistently with the public interest." However, if we recognize that the overall "public interest" incorporates both technical and political considerations, then individual decision-making becomes complex. How does the technically-oriented individual take into account that public policy considerations could outweigh technical conclusions? Is it reasonable to work on a project that is technically impossible but that is a political priority for society? How can one assure that the political decision was made with full knowledge of the technical impossibility?

Use of this Module in Teaching

This curriculum module is packaged as a PowerPoint presentation that incorporates several mpeg video clips, as outlined in the table below. (See Table II.) The complete original debate video ran over two hours, and so only the most relevant and useful portions have been digitized and extracted for use in this module. The debate presentations by Weizenbaum and Cohen are not included. Since these were the second presentation for each side of the issue, they naturally do not cover as much new material. The questions from the debate audience are also not included, as the pace of this portion of the original video is rather slow.

The complete module could easily take three 50-minute class periods, or two 75-minute class periods. With extended discussion time and/or in-class active learning exercises, covering the complete module might take an additional class period or two. On the other hand, with judicious selection of material and use of class time, the core issues might be covered in as little as one 50-minute class period. Suggestions for using the module in different formats are summarized below.

Pre-class exercises.

Students will get the most out of the module if they complete a pre-class assignment that gets them thinking about the issues. Several different possible pre-class assignments are useful. One is to have the students perform a web search to write short biographical sketches on Parnas, Seitz, and Dertouzos. At a minimum, they should discover such things as that Seitz invented the "Cosmic Cube" parallel computing architecture that gave rise to commercial systems marketed by Intel and Ncube, and that Parnas was the leader of the Naval Research Lab's "Software Cost Reduction" project (dealing with software technology in aircraft weapons systems) prior to joining the SDI computing panel. It is valuable for students to see the accomplishments of such people, and consider how the backgrounds of the debate participants qualify them to offer expert opinions on the subject. Another possible pre-class exercise is for students to go to the BMDO web site and prepare a one-page summary of the current national missile defense scenario. Yet another possibility is to have the students use Nexis or do a web search to locate information on the three to five most recent tests of missile defense system components. If the whistle-blowing aspects of the incident will be emphasized, then it will be helpful if they do some background reading ahead of time (e.g., [15, ch. 7] and a selected worksheet from that chapter).

One 50-minute class, plus homework assignment.

It should be possible to successfully use a portion of the materials to provide an overview of essential issues in testing safety-critical software in one 50-minute class. The class presentation would use about 20 PowerPoint slides, plus the video clips of Reagan, Dertouzos, and Parnas. The total time of the three video clips is about 30 minutes. This leaves just enough time to introduce the ballistic missile defense problem, present the software life cycle as an organizing framework, and orient the students to analyze Parnas' argument as a homework assignment.

The 50-minute period would be organized into three segments. The first segment would be about 10 minutes in length. It would begin with a series of a half dozen slides that support giving a basic definition of ballistic missile defense, and reminding students of the activities in the system analysis and requirements analysis phases of the traditional software life cycle. It would then move to watching the video clip of President Reagan's call for the SDI program. Based on the video clip, students are asked to formulate a high-level statement of SDI system requirements. Several students can be called on for a suggested requirements statement. The supporting powerpoint material notes that a requirements statement might focus on either of two parts of Reagan's speech. Parnas focuses on the part where Reagan says: "I call upon the scientific community to give us the means of rendering these nuclear weapons impotent and obsolete." Alternatively, Seitz focuses more on the part where Reagan says – "I am directing a long-term R&D program to begin to eliminate the threat posed by strategic nuclear missiles." In either case, the general software requirements are to take in sensor data and direct weapons systems to destroy an incoming attack before it reaches the United States.

The next segment of the class would again be about 10 minutes in length. It would mention the M.I.T.-CPSR debate, identify the participants in the debate, and then watch the six-minute video clip of Dertouzos' overview of the SDI problem. Based on his presentation, students should get a greater appreciation for the vastness of the geographic area to be monitored by sensors, the numbers of warheads and decoys to be handled in an attack, and the time scale of an attack. They should also get a better idea of the data flow and decision-making involved.

The next segment of the class would be about 30 minutes in length. The main portion of this is spent watching the video clip of Parnas' presentation. This prepares the students for a homework assignment to diagram, in premise-conclusion form, Parnas' argument. To get the students oriented for this analysis, it is useful to walk through identifying the conclusion of the argument with them. The homework assignment for the students, then, is to identify the premises used to support this technical conclusion. Students should be able to identify a sequence of three to five technical premises, and to give some indication of their own belief in the truth of each premise. The PowerPoint material includes transcribed versions of some of the overheads in Parnas' presentation. If desired, these can be printed and given to students as a handout for use in the homework assignment. The homework assignment can be handed in and graded according to how many and how well the main premises are identified. At a minimum, students should be expected to identify the premises that the specifications for the software are necessarily unknown, that there is no chance for any realistic system-level testing, and that there is no chance for debugging during operation. Additional slides can be used in a future class to review the analysis of the premises after the assignment is completed. To connect this analysis of Reagan-era SDI program with current national missile defense scenarios, students might be asked the additional homework question of how their overall analysis of the argument would change if the scenario involved no more than ten missiles and ten decoys launched from an area such as North Korea or Iraq.

The primary weakness of covering this subject in a single 50-minute class is that the "other side" of the argument, as made by Seitz, is not covered. However, Parnas advances an in-principle argument that should stand or fall on its own merits. Also, Seitz does not directly address the premises advanced by Parnas. Thus while additional time will certainly improve students' understanding of the problem, it should still be useful to cover the essentials of Parnas' argument in one 50-minute class.

One 75-minute class, plus homework assignment.

Several options are available for covering this module in one 75-minute class. One possibility is to not present any additional material from the PowerPoint and video clips, but to use the additional time for an active-learning style exercise that focuses on analyzing Parnas' argument. After watching Parnas' presentation and guiding the students to the conclusion of his technical argument, allow a short time (three to five minutes) for students to individually identify the premises supporting this conclusion. Then call on some students to give one of their premises and build a list premises on the board. Once a full premise-conclusion summary of the argument is constructed from student responses, ask for one person to argue for and another against the truth of each premise. If time permits, ask if Parnas' analogy for the level of reliability expected of SDI software (an expectation similar to that of your car starting when you turn the key) is appropriate, and if other analogies might be more appropriate. As a follow-up homework assignment, students can be asked to analyze how the truth of the premises and conclusion would change for a scenario of an attack consisting of tens of missiles from a smaller country.

A different option for one 75-minute class would be to use the material in the module to present a summary of Parnas' argument after viewing his presentation, and then watch Seitz' presentation and also use the prepared material to present a summary of his argument. The class would then end at the point where a natural homework assignment would be for students to write a short critique of the relative merits of the two arguments.

Two or more classes.

Full coverage of this module would normally take two, or possibly three, classes. This allows time to also see the video clips of the rebuttal statements, and to analyze the issues from different perspectives. It also allows time for assessment of the premises used in the arguments. An important additional perspective is to explicitly identify the ethical issues involved, and to discuss the guidance that the codes of ethics give. Students should be able to easily identify relevant items of the AITP Standards of Conduct, the ACM/IEEE-CS Software Engineering Code of Ethics, and the ACM Code of Ethics. Analysis of the guidance provided by the codes of ethics could be done either as an in-class active learning style activity or as a homework assignment.

Connection to whistle-blowing.

While Parnas' actions are commonly referred to as whistle-blowing, this case does not at all present a typical whistle-blowing scenario. If anything, this incident may have increased Parnas' professional stature and visibility. Students should not be left with the impression that the typical whistle-blower fares so well. It is important that students also see a more standard treatment of whistle blowing [15]. There are several good whistle-blowing case studies of relevance to students in computing and information systems majors. One is the case of Goodearl and Aldred versus Hughes Aircraft [13]. This case study involves (lack of) testing of hybrid computer chips used in military weapons systems. One advantage of this case is that it has been the subject of criminal and civil court cases that have run to conclusion, and so there is a good deal of documentation surrounding the case. A current case that is even more directly related to SDI is that of Nira Schwartz versus TRW [14]. In this case, an engineer working on missile defense software "has charged the company with faking tests and evaluations of a key component for the proposed $27 billion antimissile system" [14] (see also [19]). The allegations in this case can be seen to come back to the central point in Parnas' argument, that of designing a system to meet unknown specifications.

For a general introduction to whistle blowing, in particular the use of the "False Claims Act" in connection with fraud on the federal government, a good additional video resource is available from the Taxpayers Against Fraud organization [18]. The video presents short summaries of three whistle-blowing cases that involve legal action under the False Claims Act. It clearly makes the points that whistle blowing is often done at great personal cost, that it often involves saving lives as well as government money, and that it requires gathering and presenting information carefully. Importantly, the video also presents some of the history of, and motivation for, the False Claims Act (originally adopted under Abraham Lincoln). The video is just over seventeen minutes long. A short review of the video and suggestions for using it in class can be found at www.cse.nd.edu/~kwb/nsf-ufe/.

Use in a software engineering course.

When the module is used in a software engineering course, there will likely be relatively more time spent on the software testing issues and relatively less on the ethics issues. It is important that the ethics issues still be addressed, of course. At a bare minimum, students should be made aware of what the professional codes of ethics say about requirements, specifications, testing, and validation of software. Software engineering students may be able to usefully devote more time to Parnas' arguments about why SDI presents a unique computing problem and why it would not be able to be realistically tested. Also, there is a quote by James Ionson from the Reagan-era SDI office to the effect that SDI software does not have to be error-free, but only fault-tolerant, and that "if another million lines of code has to be written to ensure fault tolerance, then so be it." This quote should provide an interesting opportunity to discuss what is meant by error-free and fault-tolerant.

Use in a science, technology, and public policy course.

Students in this type of course are likely, overall, to be less interested in the technical details of software development and testing and more interested in the decision-making and public policy aspects of the case. An interesting discussion theme for this type of course may be the politics/technology decision-making conflict mentioned earlier. That is, what are the implications of making a political decision to pursue a system that is doomed to failure on technical grounds? What is the responsibility to make the technical assessment of the project known to the public? What it the responsibility of technical professionals working on such a project – does pursuit of quality standards still have meaning?

Challenging Real-World Problem

Safety-critical software is an important topic for courses in ethics and computing, computers and society, software engineering, technology and public policy, and other related areas. The missile defense problem presents the most challenging real-world software engineering problem imaginable – to interpret real-time sensor data taken under natural conditions and appropriately handle an attack by an intelligent adversary likely to employ strategies that have not been fully anticipated. The currency of the national missile defense problem makes analysis of this Reagan-era SDI case study highly relevant for today's students. The historical view of over fifteen years should allow a more objective evaluation of the issues. The basic technical issues still apply to any system envisioned today.

This case study allows opportunities for extended critical-thinking exercises, including the development of summary pro/con arguments and the design and evaluation of system testing plans. It also allows opportunity for analysis of how the professional codes of ethics deal with the issues involved, and connection to whistle-blowing topics. For advanced students in computing majors, it can be used to provide motivation for discussion of concepts such as fault-tolerance in software, consistency in distributed databases, and the Byzantine agreement problem.

This curriculum module is being made available free of charge for use in academic teaching. The materials may be down-loaded from the web site http://www.cse.nd.edu/~kwb/nsf-ufe/starwars/. This web site also contains a wealth of other materials created under partial sponsorship of an NSF DUE grant on teaching ethics and computing. Also, faculty may obtain a copy of the material by sending two blank CDs to the author, with stamped, self-addressed return mailing container.

Acknowledgment

Special thanks are due to David Parnas and Chuck Seitz for reading and commenting on earlier drafts of this paper. It was Chuck Seitz' suggestion to include copies of the Eastport Report and the OTA report with the other materials for the module.

Professors Robin Murphy and Bill Albrecht at USF, Don Gotterbarn at ETSU, Doris Appleby at Marymount College, and Gordon Hull at Vanderbilt provided valuable feedback from classroom-testing an early draft of this module, resulting in a number of improvements. Thanks are also due to an anonymous reviewer who provided several excellent suggestions for revisions that have been incorporated into the final version.

Christine Kranenburg and Laura Malave provided substantial assistance in creating the PowerPoint and digitized video to support the use of this module. The idea to develop a curriculum module on this topic was originally suggested to me by Joe Wujek at one of the NSF-sponsored UFE workshops.

References

[1] R. Reagan, "Address to the nation on national security," Mar. 23, 1983, VHS video, The Reagan Library, 40 Presidential Drive, Simi Valley California, 93065-0699. http://www.reagan.utexas.edu/.

[2] D.L. Parnas, "Software aspects of strategic defense systems," *Communications of the ACM*, vol. 28, no. 12, pp. 1332-1335, Dec. 1985.

[3] D.E. Mosher, "The grand plans," *IEEE Spectrum*, vol. 34, no. 9, pp. 28-39, Sept. 1997.

[4] Special issue on ballistic missile defense, *IEEE Spectrum*, vol. 34, no. 9, Sept. 1997.

[5] The Association of Information Technology Professionals (AITP). Web site http://www.aitp.org.

[6] DOD Ballistic Missile Defense Organization (BMDO). Web site http://www.acq.osd.mil/bmdo/.

[7] W. Panofsky, "Nuclear offense versus defense," *Science*, vol. 291, no. 23, Feb. 2001, 1447.

[8] D.L. Parnas, "Parnas on Parnas: A life of indecision," *ACM SigSoft Software Engineering Notes,* vol. 24, no. 4, pp. 47-49, July 1999.

[9] W.J. Broad, "Scientist at work: Philip E. Coyle III; Words of caution on missile defense," *New York Times*, Jan. 16, 2001.

[10] R.J. Smith, "Bad weather, computer woes delay laser test," *The Washington Post*, Oct. 8, 1997.

[11] "Possible Soviet responses to the U.S. Strategic Defense Initiative," Central Intelligence Agency memo NICM 83-10017, Sept. 12, 1983. Available at http://www.fas.org/spp/starwars/offdocs/m8310017.htm

[12] "The SDI imperative" (editorial), *National Review*, Feb. 22, 1999.

[13] K.W. Bowyer, "Goodearl and Aldred versus Hughes Aircraft: A whistle-blowing case study," *Frontiers in Education* (FIE '00), pp. S2F-2-S2F-7, Oct. 2000.

[14] "Former engineer says company faked tests," *The Tampa Tribune*, Mar. 7, 2000.

[15] K.W. Bowyer, *Ethics and Computing: Living Responsibly In A Computerized World,* 2nd ed. New York, NY: IEEE/Wiley, 2001.

[16] K.W. Bowyer, "Resources for teaching ethics and computing," *J. Information Systems Education,* vol. 11, no. 3-4, pp. 91-92, Summer-Fall 2000.

[17] Software Engineering Code of Ethics, IEEE Computer Society web site: http://www.computer.org.

[18] *Taxpayers Against Fraud, Fighting Fraud: Citizen Action and the Qui Tam Remedy*, VHS format video tape can be ordered from www.taf.org. Taxpayers Against Fraud, The False Claims Act Legal Center / 1220 19th Street, NW, Suite 501 / Washington, DC 20036.

[19] 60 Minutes II, America's Dream Defense, originally aired Dec. 26, 2000. Transcript available from CBS News through Burrell's Information Services. 1-800-777-8398.

[20] K.W. Bowyer, " 'Star Wars' revisited – A continuing case study in ethics and safety-critical software," in *Proc. Int. Symp. Technology and Society 2001 (ISTAS '01)*, July 2001. A shorter version also appears in Frontiers in Education 2001 (FIE '01).

[21] F.P. Brooks, *The Mythical Man-Month*. Reading, MA: Addison-Wesley, 1995.

[22] Eastport Study Group, Summer Study 1985: Rep. to the SDIO Director, Dec. 1985.

[23] U.S. Congress, Office of Technology Assessment*, Ballistic Missile Defense Technologies*, OTA-ISC-254. Washington, DC: U.S. Government Printing Office, Sept. 1985.

[24] "U.S. may deploy defenses untested," *Tampa Tribune,* June 8, 2001.

[25] "Bush missile plan faces huge obstacle," *Tampa Tribune,* June 9, 2001.

[26] Comments by Newt Gingrich in the transcript of National Public Radio's "All Things Considered," July 18, 2001. Available through http://www.npr.org/about/transcripts/index.html