Chapter 7

Programmers and Systems Analysts

Ethical Dilemmas in this Chapter:

- Coding Practices
- Code Maintenance
- Code Review
- Code Design and Testing
- Programmers and Viruses
- Programmer Security Responsibility

Introduction

This chapter covers the ethical issues that programmers and analysts may encounter. We discuss the ethical dilemmas resulting from coding practices, reviewing code, malicious attacks, secure programming, software deployment, code design, and program testing.

We start by defining the roles of programmers and analysts. Programmers and analysts are the information technologists who develop computer applications utilizing various programming languages and macros.

In this chapter, we refer to programmers and system analysts in various ways depending on the role they are playing, including application and software developers. It is important to know that they all perform the same information technology (IT) role with slight variations. For example, a software developer is a programmer that focuses on the creation of a software product that can then be packaged and sold. A systems analyst in some cases is someone who writes code for the operating system such as UNIX shell scripting.

Programmers work off of design specifications. Specifications are detailed instructions written by business analysts or system users, which define the requirements for a given application slotted for development. The programmer takes these requirements and converts them to a program design.

There are four primary phases of software/application development: design, plan, prototype, and implement. If any of these steps are skipped, a problem may develop. This chapter touches on and provides scenarios of ethical matters for each phase of the software life cycle development process. We also discuss common programmer ethical dilemmas.

Coding Practices

This section discusses ethical quandaries regarding bad code, weak code, the correct use of system memory, utilizing system resources, redoing code due to business rule changes, staying current with coding best practices, and pseudo code. This will give you a thorough understanding of the ethical issues programmer's face in their general coding practices.

The primary ethical concerns of programmers result from either lack of proper communication between business users and developers, insufficient knowledge in their proclaimed area of programming expertise, simple boredom or laziness, and most common of all, a lack of time allocated for proper project completion.

Bad Code – Whose Problem is It?

Bad code is code that is written in a manner that does not fulfill the business or technical requirements defined by any given project. Writing bad code can mean several things. Sometimes it is just a matter of not taking fully into account the complexity of the specifications. At other times, developers are simply working off of patterns that they have used successfully in the past that may not necessarily apply to the current project. Bad code can also result from rushed deadlines and production pressures. Finally, and worst of all, bad code can come from developers who do not have the technical expertise to get the job done, but somehow managed to bluff their way through the technical interview. The choice of how to handle them falls into the hands of the programmer. The following scenario addresses reworking bad code that was written by a coworker and friend:

You are performing software development with a computer graphics group. The team is motivated to produce an award-winning product for the next release. However, one member of the team, "Allen," produces dubious code and seems incapable of doing better. The project schedule is in serious jeopardy. One of your fellow programmers, "Rick," suggests that you and he just rewrite the code written by the weak team member Allen, so it works properly. He says that no one need ever know that the two of you rewrote Allen's code to make it work properly. Is this type of cover up performance ethical?

Conservative Rewriting programs for someone else on your team is inappropriate and slows production down, wasting your time and the company's time. You should sit down with management and explain that this one person is significantly affecting the goals of the department and company. You should never do someone else's work for them, no matter how well meaning your intentions.

Liberal Your desire to help the team and your friend is well intended and may work as a one-time fix in this case. There is nothing wrong with helping a friend out. You need to watch each other's backs. You may need help sometime in the future and he will owe you one.

In addition, you may also want to consider talking to Allen and encouraging him to brush up on his programming skills so that he is up to speed. For Allen's benefit, you should also tell him that you are making changes to his code. This will ensure that he can respond appropriately if there are any questions directed to him in the future or any program maintenance work that comes back to him. It will also help him learn how he should have written the code.

SUMMARY

It is a tricky business to help someone do their work without notifying management. Expectations are set around a person's performance; in this situation you will create a false sense of Allen's ability. If you really want to fix the code, you should seek permission from the project leader to do so.

Working on a team brings to light many ethical issues. When you cover up for someone else's failings, it can put the team and project at risk. However, there also needs to be a human element in business.

No one is perfect and any expectation of that is unrealistic. Most people find it better to work in an environment where they feel supported rather than in a cutthroat environment where everyone plays the blame game. How far you take that support is up to you. In addition, how often you point the finger is up to you as well.

SOAP

Microsoft Windows NT

Does it infuriate you when a PC user calls and says that their machine is running slow? The first thoughts that come to mind would be a memory issue or a network card problem, if hooked to a network, or disk space problem. But after investigation, you find that the real problem is Windows NT's baseline OS doesn't include defrag software. The end user has to buy something like Disk Keeper to rid the hard drive of severe fragmentation. Also, even with Disk Keeper you have to run the software repeatedly and it still doesn't fully rid the hard drive of fragmentation. Yes, Windows NT was far better from a security standpoint than Windows 95/98, but get real; even these baseline operating systems included some form of defrag software. It would be like using all your money to buy what you thought was reliable transportation but it couldn't even get you out of your driveway.

Jeff Payne
GIAC Certified
IA Analyst Inc.

Weak Code – Is it Ever Okay?

Weak code falls into the category of laziness, carelessness, or results from overworking a developer. In the following circumstance, weak coding practices are due to being overworked, which commonly occurs in application development environments that have unrealistic deadlines generated by the business users or management. In the following two scenarios, consider how you would respond to weak code.

You are tired from working a week of consecutive ten-hour days to get your project done and decide to cut corners wherever you can, which results in writing lazy code that will just get by. The code you end up writing will work fine and does not have any security weaknesses, but you know you could write it much better and utilize computing resources better. Is it okay to write weak code when you do not have the time to do the job better because you are being grossly overworked?

Conservative Your company pays you good money to create the optimum solution, not the minimum optimum solution, for your development efforts. Cutting corners never pays off and you will probably have to come back and rewrite your weak code anyway. Dig deep and do the job correctly, even if you are tired. You will be glad you did. Nothing feels better then knowing you did a job to the best of your ability.

Liberal When push comes to shove, it does not make any difference what kind of code you write as long as it gets the job done effectively, within budget, and on time; that is the bottom line for management. That is what you are being paid for, not to create a code masterpiece. There is no need to spend three hours working on code that you could write in one just to make it fully optimized. If you always did that, you would never get any projects done on time.

SUMMARY

In all areas of business, we are faced with the dilemma of doing the job versus doing the job to the best of our ability. Doing the job right tends to pay off in the end, but not always if you miss your deadline. Finding an ethical balance that you can live with between giving it your all and pulling projects in under deadline will give you peace of mind in the work place.

Correct Use of Memory – Fixed Versus Dynamic

Memory allocation is an important matter for lower level programmers such as C, Java, and UNIX developers. Properly using system memory can be an ethical issue because wasting memory costs the IT department money and can interfere with other developers' work. Consider the following scenario:

You are developing a program for an accounting firm, which draws data from an accounts database table. The size of the data you extract may vary depending on the records for any given month. Is it unethical or just poor coding practice to waste memory by declaring the maximum possible size rather than dynamic allocation?

Conservative If you are wasting memory, you are wasting system resources, which results in wasting the company's money. Any act of wasting money is unethical. How would you feel if it was your money and someone was not handling it with utmost care and respect? You would not want someone else to waste your hard-earned cash.

Liberal In this case, maybe you did not choose the most optimum programming solution, but this is really no big deal compared to today's hardware system resources. We are talking about worrying about a fraction of a cent here. If every programmer had to worry about these matters, no work would ever be done.

SUMMARY

Programmers are hired for their expertise in developing software. The allocation and use of memory is entrusted to them. If they remain within a certain range of acceptability, this should not become an issue in most cases. Everyone takes shortcuts from time to time. Many of these shortcuts are generated because the developers are under aggressive and sometimes impossible deadlines. Other times, these shortcuts are bad habits that they have developed over time. In either case, it is worthwhile to consider both extremes and find out what you feel comfortable with as an application developer. Maybe you have never considered how you waste money through your coding practices. It is worth some thought.

Ethical Use of System Resources – Are You Using or Abusing the Privilege?

You are a computer genius working at a securities company and can write code that will bump up the speed of the host computer while it is performing complex trading calculations. Is it ethically appropriate to alter the speed of a computer, which will be running your application?

Conservative You need to be very careful when altering the system resources of a machine that will run your software. It is not recommended to reallocate the fundamental core speed of host machines. This practice is unacceptable due to the risk imposed on the host machine. Think about the fact that if there is the slightest problem or security weakness in the software that you have developed, you can impose serious damage to the host computer. Let the client upgrade their hardware on their own. Do not interfere with this type of dynamic memory boost, even for the client's benefit.

Liberal If you know how to write code that speeds up the processing power of the host machine, you should review it with your technical project leader and, if they approve, go for it. This will give your product a significant edge on all of your competitors. The clients will most certainly appreciate the additional speed and processing power. Just be sure you thoroughly perform error handling and penetration testing on the product before it is released to the general population.

SUMMARY

Altering system resources on a host machine is not against the law or necessarily immoral, but it could result in problems, depending on the structural integrity of the software manipulating the client resources. You will see the negative results of this ethical dilemma if your software has a bug that permanently alters a host machine or makes is susceptible to a malicious attack.

Redoing Code Due to Management Changes – Should You do It?

Often on IT projects, the technical team is required to get started prior to when the management team has actually completed the full project planning. Sometimes management will notify the programmers that the coding standards or even the planned software project itself has changed to implement new business rules. This type of business requirement change results in reworking code and sometimes tossing out weeks or months of development work. The following issue addresses this problem.

You are halfway through a large program that will be used by the order fulfillment department of a fiber-optics company. Your IT manager implements new standards for everyone to follow. Do you go back and rework everything you did? Do you start the standards from the point you are at? Do you say forget it and use them on the next project?

Conservative If management has provided you with standards, it is your responsibility to utilize them to the best of your ability. Go back and try to update the code you have already written to meet the current standards. If you cannot update it, you must rewrite it. This is part of the job. Roll with the punches. Be sure you speak to your manager so that they know that reworking and rewriting your earlier work will set the project back in terms of the aggressive deadline.

Liberal Just because management did not perform their job correctly is no reason for you to suffer the consequences. After all, if the project is not completed on time because of this change, you will be the one to eat it, not management. Just implement the standards from the point you are at in development and do not worry about the work you have already done. You will never get the project done on time if you back track at this point.

In addition, if implementing the new standards throws the project off too much because you cannot use existing libraries, do not implement the standards at all. It is not your fault they were tossed into the mix at this late phase of the development effort.

Summary

One constant in application development is that things are always changing. Operating systems are upgraded that require reworking old code, business rules change, and IT projects are often dumped or redesigned in mid-development. Learning to adapt to change is essential for programmers and analysts.

Staying Current with Coding Practices – What if It's Not in Your Schedule?

Most of your development team has remained current with recent coding languages, practices, standards, and changes in the development industry. You do not have time for the extra education required to stay current. Do you feel it is ethically correct to remain in the dark when it comes to the current coding standards and practices?

Conservative Part of a programmer's job is to stay current with the most recent developments in IT. This includes educating yourself on the most up-to-date compilers, programming techniques, secure coding practices, and standards. It is a given in the development industry that you will grant yourself a continuing education in your job role and potentially expand beyond your area of expertise. Therefore, it is not acceptable if you fail to keep yourself informed of the best and current means to perform your job as an application developer.

Liberal If your company does not pay you to take educational seminars and provide you with up-to-date resources, it is not your responsibility to do so on your own. Granted, you want to keep your skills fresh and marketable, but there is only so much time in the day to realistically do this. Educational seminars can be costly if you have to foot the bill yourself.

Spending all of your time working or in school will consume your life. That is not a fair or a happy way to live.

SUMMARY

Keeping up to date on recent software tools and development practices requires a significant amount of time on the part of the programmer. Some programmers do this automatically because they are excited about new technology in their area of expertise. Others may simply not have the time because they carry a large workload at the office. A third category of programmers may simply not want to update their skills and coast for as long as they can. Find out what type of programmer you are and if you feel good about it.

Commenting Code – When is It Okay Not to Do It?

The practice of commenting code is important for the other developers who will later maintain, fix, update, or rewrite the code you create. Commenting practices vary from developer to developer. Some programmers hate commenting, and therefore never do it. Others want to make themselves indispensable and do not comment because they want the work returned to them. Others comment consistently. Consider the following scenario:

You are a top coder at your company and you never comment your code because it takes too much time. This is primarily to keep up with what you are required to produce. Is it unethical to fail to comment code due to a lack of time?

Conservative Most programming standards dictate that developers must provide code comments. This is sensible because it allows for easier updating of the program and helps others understand what the initial developer was doing. Programmers should make commenting code a standard duty. If they do not, they are intentionally making it harder for those that will maintain their code. Saying that it takes too much time is not an excuse. In the long run, it will waste time for the business.

Liberal The bulk of programmers do not comment their code. In most cases, if there is a problem with code a developer has written, they will fix it themselves. When faced with the choice to finish a project on time or comment the code and run over schedule, the wise choice is to get the job done quickly.

SUMMARY

Aggressive deadlines in development result in an array of ethical dilemmas. The above example is one such circumstance. It is always best to weigh the advantages against the disadvantages when making a determination such as commenting code.

Omitting Code Comments for Job Security – Should You Play the Game?

You are a top coder at your company, and you never comment your code because you want to ensure job security by making it difficult for others to understand. Is it unethical to avoid commenting your code for the purposes of job security?

Conservative Failing to comment code to ensure job security is definitely not ethical. It is also a form of sabotage. Making things hard for the company and team due to your own insecurity is just wrong. Any practice that reduces the quality of your work to gain job security is unethical.

Liberal Office politics are tricky and sometimes you do not have a choice but to play the game. If you do not play the game, you usually lose. The ethical people are not always the ones on top. Unfortunately, it is usually the reverse. People who know how to play the game are the individuals calling the shots. Therefore, if you write excellent code, it is smart in some cases to ensure your job security by making it a little harder for others to decipher your work. Additionally, in this way no one else can take credit for your work because they would be required to speak to you in order to understand it.

SUMMARY

Job security is a very interesting ethical matter. Most people build a little job security into their jobs. Some people in IT are terribly bad at playing politics so they use their technical knowledge as a non-political strategy to stay in the game.

Pseudo Code – Is It Worth Your Time?

Pseudo code is the step-by-step description representing future code that will be written in its place. Pseudo code is not often practiced outside of the computer school environment. Take into account the following scenario:

You never pseudo code even though it is company policy. A non-technical manager using methodology books wrote the company coding policy and most of the technical people think the corporate policy makes no practical sense. You think pseudo coding is a complete waste of time, and therefore, you are saving your company time and money by not pseudo coding. In this case, is it appropriate to go against company policy?

Conservative In most cases, you should follow the standards set by the company. Standards are written by experts and are tried and true. In a few cases, it is useful to address unnecessary standards with your manager. You should sit down with your manager and explain the pros and cons of pseudo code. Tell them that you can save the company time and money by not adhering to this standard. Explain why it is unimportant technically and that it will not affect the quality of your work.

Liberal There is no need to make a big deal out of a standard such as pseudo code, which no one in the industry does anymore. Do not pseudo code. There are plenty of standards that are not followed, because the development team thinks the standards document is a joke. Most programmers have honestly not even reviewed the standards because they did not want to waste their time. At least you know what they are.

This is a case of realism versus textbook (at best). In theory, all kinds of standards and practices are important, but in real life, development efforts with real deadlines and most standards and methodologies do not apply. This is especially true if the standards were written by non-technical personnel.

SUMMARY

Often, developers are faced with long drawn out methodologies and standards that, if followed, would prevent all projects from ever being completed. Given enough time to elaborate standards and methodologies is beneficial to software development, but with the deadlines imposed on programmers they are forced to find the best possible solution somewhere in the middle. Senior management does not always understand this.

Code Maintenance

Maintaining code poses different ethical situations than the original generation of code. Programmers who maintain code have the job of going through existing code and making minor adjustments due to changes in the production environment, database, business rules, or overall system architecture. The next two issues involve modifying the original code of a program.

Modifying the Original Secure Design of a Program – Is this Ever Appropriate?

You are a maintenance programmer working on a program that has a security model with bounds checking. You add a new function that does not perform bounds checking. Is it ethical for a maintenance coder to modify the original secure design of a program in this manner?

Conservative When performing maintenance work you must stick with the existing design and security model of a given program. It is never appropriate to add functionality that does not adhere to the original secure design of the system. You are wasting your and the company's time.

Liberal It is not always necessary to following existing secure design concepts of a program when you are simply adding functionality. The overall secure structure of the program should be enough to maintain its own integrity, even if you do not add bounds checking to your new function.

SUMMARY

Adhering to a program's secure structure when you are doing maintenance work is primarily dependent on the type of maintenance work you are doing. If the new function overrides or alters the functionality of the program altogether and you do not implement bounds checking, you may be ethically wrong. However, if the additional function you are adding has no effect on the overall security of the program, there may not be a need to address it.

Affecting the Overall Quality of a Program – What is Acceptable?

Is it ethical for a maintenance coder to write a less quality add-on function than the code written for the original program?

Conservative Continuation of quality is a requirement for maintenance programming. Any additional code added to a program should have the same level of quality.

Liberal The reality is that compensation for maintenance programmers is far less than for the original developers, because they are often not as skilled. Trying to match the quality of an existing program may not be realistic for a maintenance programmer.

SUMMARY

Quality is important in the maintenance of existing applications; however, the resources on hand must be taken into account and an acceptable quality assurance level must be set. Not all programmers are phenomenal, and to set a standard for all work to match may be unrealistic. However, system security should never be the aspect of maintenance code that is comprised for any reason.

Code Review

Thus far in this chapter we have discussed standards and methodologies from the point of view of the programmer. Now let's take a look at it from the point of view of the reviewer role.

The code review process occurs in three different ways: by a third party, by coworkers, or by automated software. The process of reviewing code provides a check and balance for development efforts. A simple code review checks the basic vulnerabilities of the program. The more complex security conscious code reviews are referred to as penetration tests. These tests probe and attempt to penetrate the security of the program and system in which it is running.

This section discusses lazy reviews, following standards, and automated code review.

Lazy Reviews – Do You Do It?

Your team takes turns reviewing each other's code on a monthly basis. You know everyone writes decent code. Is it acceptable not to perform the review of your team's work if you do not have the time?

Conservative Absolutely not! A code review checks for more than just good coding practices, it verifies the integrity and security of the system. No one writes perfect code. It is important for every developer to have a second set of eyes to review their work.

Liberal If your coworker has just created routine code that they could write blindfolded, taking the extra time to review it when you already know it is fine is a waste of time and energy. This is especially true when you have so many other projects to focus on.

SUMMARY

Getting lazy reviewing code or just feeling it is not necessary could save time or cause real damage. The complexity and priority of the project that requires review plays a role in the necessity of a thorough review. When the code to review is repetitive, maybe it is a waste of time to check every line. But then again maybe not, that is for you to decide under the unique circumstances of your job as programmer and code reviewer.

Following Standards – How Strict Should You Be?

When reviewing code, you discover that one of the programmers did not follow the coding standards but otherwise their work is good. Do you make them rewrite it or let it slide knowing it may cause someone else more work in the future?

Conservative Your job as the code reviewer is to make sure applications meet coding and security best practices. If the programmer knowingly failed to implement company standards, they should expect to be required to rewrite it after the code review. If you let one programmer get away with it, they all will try and you will end up with quite a mess on your hands.

Liberal If the code is good and secure, there is no need to waste time and money making the programmer rewrite it on principle. Instead, tell them in a friendly way that you let it slide since it was such excellent code, but they should review the programming standards set up by the department for future use. There is no need to make a big deal of it, but have them try to make the adjustment for the next project they work on.

SUMMARY

When deciding to enforce standards on programmers after a code review, consider if the time and energy required will justify the changes in the code. If so, require the programmers to go back and redo their work to the department standards. If not, you may be slowing down the entire project and creating a lot of enemies. Remember, the goal of software development is to create the best possible program in a time efficient manner. The only absolute requirements are that the program performs operations according to specification and that it does not pose a security risk to the system.

Automated Code Review – Do You Trust the Software or Yourself?

Automated code review includes the use of software applications to review programs. There are many tools on the market that perform this function. The following example discusses the ethical circumstance of relying exclusively on automated code review. For instance, you use an automated code review tool to review your own Visual Basic code. When you run the software on the code you just built, it detects structural problems that will take you weeks to correct. You are an expert in Visual Basic programming and personally do not trust automated code reviews so you do not update your code accordingly. Did you do something wrong?

Conservative The purpose of an automated code review tool is to more quickly find mistakes you would not otherwise find through a personal or peer review. Research the recommended changes and implement them. It is never appropriate to ignore the recommendations of any type of code review.

Liberal You have determined that automated code reviews can be generic and trust your expertise in Visual Basic programming. Why waste the time and energy in implementing a structural change that is not necessary. The automated tool may only be programmed to find one type of structure and therefore be incorrect itself.

SUMMARY

Implementing an automated code review and following up on that review with changes to your code has strengths and weaknesses. If you know that the structural changes the tool is recommending are irrelevant you may decide to skip them. However, on the other hand, if you do not know as much as you think you do and fail to implement changes in your code based on the review, you may have a lot of extra work to do in the future as well as set the project off of its deadline.

Code Design and Testing

The code design and testing phases of application development occur at opposite ends of the life cycle development process. However, they are discussed jointly in this section because they are codependent. Some IT departments perform testing instead of code design and then rework the program at the testing phase. This type of coding is known as "ad hoc development." Other IT departments perform ample code design and thus have minimum reworking to do at the testing phase.

The subjects discussed in this section are skipping the design phase, ad hoc development, and skimping on the test phase.

Skipping the Design Phase – A Bad Idea?

The design phase of programming is the structural program development. This is where you lay out the process flow and overall feasibility.

You decide to skip a thorough design phase for your system because you hate designing and love coding. Once you get to the testing phase, there are major structural problems but the code is great. Do you go back and restructure it or just write workaround code to make it usable?

Conservative You should have done the job correctly from the start of your development efforts. Now that you have determined that skipping the code design phase really does not work, you should go back and perform a code design and rewrite your code so that it works correctly with a strong architecture behind it.

Liberal It is too late in the game now to go back and rework the design of the application. Write the appropriate workarounds that will make your application function correctly. Based on this experience, you may want to consider taking the time in your next development effort to work the kinks out at the time of code design.

SUMMARY

Skipping phases of the software life cycle development process always comes at a price. Unfortunately, the above issue is quite common in software development. It tends to occur with the most talented computer programmers who feel that they are beyond design.

Ad Hoc Development – Is a Structured Plan Better?

Most organizations do not have time to fully implement a structured development plan that utilizes best practices. Therefore, the unofficial standard at some companies in regard to application development is ad hoc coding and then deployment.

Your company is one such organization that develops on an ad hoc basis. You know that this method of application development does not fulfill even the most basic application system requirements. What do you do in this circumstance?

Conservative You should take the initiative to propose a full life cycle development process to your manager and IT team. Writing code under the current circumstances of your job will only generate future problems. You will be thanked for your foresight.

Liberal Unfortunately, many development efforts occur in an ad hoc manner. If you are working in an environment like this, you may choose to go with the flow or set up a process for yourself to use that is more structured. Proposing an entirely different means for development efforts may set you at odds with the entire team and probably will not be implemented anyway.

SUMMARY

When you work for a company that performs ad hoc application development, you have the personal ethical choice to implement a more structured means for your own development process. You also have the choice to take it to the next level and share your process with the rest of the team and/or management.

Skimping on the Testing Phase – Is Automated Testing Enough?

The primary purpose for testing an application prior to its entry into production is to verify security. Skimping on the testing phase can result in exaggerated problems for the future of the application. Most departments do not spend enough time testing applications prior to placing them in production.

Sometimes this happens because the developer is used to their code and confident in what they have written. Thus, they feel it is not necessary to test it. Other times they just check for obvious errors and pass the application to the deployment team. For these reasons, it is always optimum to have a different person perform testing on a given application.

You have just completed an application that is a month overdue. You use an automated testing tool to verify that it does not have any major bugs or security weaknesses. Is automated testing enough or are you ethically required to perform full testing?

Conservative Automated testing does not necessarily cover all of the bases of human testing. It may cover different areas that human testing misses. To properly complement automated testing, you should now perform a peer test of the application. Have a coworker test the system for you.

Liberal Once you have completed automated testing, you have done your job according to the company standards. It is not your responsibility to enforce additional testing if that is not the company policy. It is not even realistic to perform that testing given that the application was supposed to be deployed one month ago.

SUMMARY

Going above and beyond the call of duty to ensure that your application is as good as required is a personal decision. In most IT departments, you will find a range of personnel from those who do the minimum just to get by to those who are so thorough they slow the entire process down. Finding the right balance of thoroughness and expediency is important to a successful development process.

Programmers and Viruses

The intentional and unintentional introduction of viruses into development and production environments requires significant ethical consideration. Viruses are sometimes propagated through a development environment due to carelessness. Programmers populate them in the production environment, when an application either intentionally or unintentionally introduces them.

The following issues, including the correct use of company resources, propagating viruses on the corporate network, downloading shareware, and malicious attacks, ethically illustrate this concern.

Using Company Resources for Continuing Education – What About the Risk?

The most common place to find computer viruses is a university technical lab. The following issue addresses the ethical concerns of contracting a virus on your corporate laptop by using company resources in a university lab. For instance:

You are a junior programmer at a start up company. You are attending night school to finish your education in computer science. You use your company laptop to perform homework assignments at the university lab, which puts the laptop at risk. Is it appropriate to put company resources at risk in such away?

Conservative Even if you are pressed to use company resources for your continuing education, it is not appropriate. Company resources are for company use only. Any use of the corporate laptop outside of company use is not acceptable.

Liberal Since you are using your company's laptop to further your education in computer science, which will in turn benefit your company, this type of use is acceptable. It would be different if you were using it for other means.

SUMMARY

Using your corporate laptop outside of work is a sensitive matter. Many businesses allow their developers to take their laptops home, especially when the workload is large and they will be doing additional coding outside of the office. The acceptable range of personal use for corporate laptops varies from business to business.

Contracting a Virus and Propagating It to the Network – Should You Confess?

This next issue follows up on the previous issue of the use of corporate resources. The only difference is that in this case you have contracted a virus from the university computer lab:

In addition to using company resources for your continuing education, you unknowingly contract a computer virus. This occurred while you were working in the computer lab at the university, which you know is prone to computer viruses and worms. The virus proceeds to infect the entire network at work. Do you tell your boss how you got it?

Conservative Taking your work computer to a computer lab at a university when you know you are placing it at risk is just plain dumb and ethically inappropriate. University computer labs are known for having every virus imaginable and you were not ignorant to that fact. You made a huge mistake.

Tell your boss what you have done and any details you can remember that may help them combat the virus, which has now attacked the corporate network. Pray that you do not cause too much damage.

Liberal Wait and see what happens. Maybe it is a simple problem and will be handled quickly. If the computer virus seems to be complex, talk to your teammates who are trying to fix it and explain what happened with your machine. Maybe that will give them the knowledge they need to clean up the virus without getting your boss into it.

Finally, reconsider whether you should go back to the university computer lab in the future. It might not be worth your job.

SUMMARY

Admitting to propagating a virus on your corporate network can be scary business. This is especially true if you have used company resources potentially inappropriately. It is always best to tell someone if you know how the virus was propagated, because it will help them fix the program more quickly.

Downloading Shareware – Is There a Standard Policy?

A lot of programmers use shareware to obtain code that they would otherwise have to spend a significant amount of time developing.

A programmer downloads shareware code from a non-trusted source to save time developing the same code. They fail to share this action with the Information Security Manager. Is this inappropriate behavior on the part of the programmer?

Conservative This is highly inappropriate behavior and can affect the integrity of the entire system. All shareware code downloads should be discussed with the Information Security Manager prior to any download. Your company may also have a standard policy to not use shareware. This action was done in ignorance.

Liberal If you have to run everything by the Information Security Manager, you will never get any work done. There must be some trust for you as a programmer, that you are knowledgeable about what you are downloading. Programmers use endless shareware libraries. This is standard practice in application development.

SUMMARY

Downloading shareware opens an entire box of ethical issues. In the one addressed above, the programmer and the Information Security Manager should set up a shareware standard in advance. If the Information Security Manager is too stringent, the programmer will have difficulty getting the job at hand done. This will consequently result in an uncooperative relationship between the programmer and the Information Security Manager.

Coding Attacks – Are They Ever Justified?

A malicious attack is designed to disable, obtain information from, or damage a computer system. In terms of coding, the goal of an attack is to destroy or derive information from a system. Most attacks result in an effect on the business, which can have a negative impact on the image of the company or availability of services rendered. There two types of coding attacks that occur at different phases of the application life cycle: architectural/design level attacks and implementation attacks.

The more difficult of the two to resolve are architectural and design. The next level of attack is the implementation level. Implementation level attacks occur at the coding stage. The code level vulnerabilities are the simplest type to repair. What would you do in this next scenario?

You have a grievance with the company you are working for that has treated you unfairly. Knowing that architectural level attacks are hard to find, you build one into the system you are coding. Are you justified in this type of behavior?

Conservative It is never appropriate to perform any type of malicious attack regardless of how unfairly you have been treated. In most cases, this type of behavior moves you out of the ethical arena into the illegal.

Liberal Business is war. If you have been severely mistreated and have no other recourse to take, you may consider this type of behavior. However, with malicious attacks you must understand that you are venturing out of the ethical behavior dilemma and into the illegal.

SUMMARY

Generating a malicious attack against a company you work for is dangerous and not a very smart idea in general. Putting ethics aside, placing yourself in greater risk by toying with the idea of a malicious attack results in much worse consequences then just unfair treatment at the office.

Programmer Security Responsibility

Determining the development security responsibility of a programmer is no small matter. Information security touches many areas of application development including application program interface (API) calls, Common Gateway Interface (CGI) flaws, temporary fixes, disabling system warnings in macro development, and using back doors. The following examples address each of these issues.

The Power of the Programmer

A programmer or a systems analyst effectively has more power than any other user of computer systems. Programmers, through code, can change the behavior of an operating system, modify the way other programs work, or even change the way system hardware is configured or functions. Overall, that is a very large responsibility to have.

So how do most programmers relieve the stress of responsibly wielding this power or the stress of short deadlines and impossible project goals? Easter eggs! An Easter egg is a little 'surprise' that a programmer has hidden within an application. It's usually a fairly small amount of code the displays the programmer's name in a unique way or causes the program to act slightly differently than originally intended.

Are Easter eggs ethical? After all, they consume system resources, take additional coding work, and are generally not bug tested as well as 'official' code. Look at it like this… Considering the power a programmer wields and the things a programmer could do to a system, adding a couple of Easter eggs is pretty harmless. I would much rather see the name of the programmer displayed in twelve different fonts than have my CD-ROM suddenly stop playing any CDs except for Pink Floyd music CDs.

Jeremy Faircloth
Systems Engineer and Author

API Calls and Security – Finding the Right Balance

You are writing code that calls functions from the Windows API.You just assume that the API is secure. Is it your responsibility to check this fact out and build additional security into your error-handling routines if the Windows API does not meet your company's information security standards?

Conservative When developing code it is best not to assume anything. Therefore, in the above circumstance, perform the necessary research to ensure that the calls you are making to the Windows API will be secure on both fronts by determining whether the API itself is completely secure.

Liberal It is not your responsibility to check every Windows API function to determine if it has the appropriate security and prevents risk. This would result in an excessive amount of time spent on research that may not be accurate anyway.

SUMMARY

When you are utilizing external interfaces and tools, you are posed with potential security risks. If you take the time to check every external function or library you may never get your code written. Finding a proper balance between getting your work done and ensuring security is up to each software developer.

CGI Flaws – Who is Responsible?

A CGI is a predefined standard used for interfacing World Wide Web (WWW) information and database servers with external software applications. CGI programs are common in Internet development work. You are writing CGI code for a Web application in the perl programming language. You were not aware of a corporate policy that involves checking input buffer lengths that can create a security vulnerability. Consequently, the corporate Web server is attacked. You wrote excellent code but did not properly check the input buffer lengths. Is the attack your fault?

Conservative It is your responsibility as a programmer to familiarize yourself with corporate standards, security practices, and policy. Therefore, you are to blame for the security breach of the corporate Web server.

Liberal Many corporations do not fully communicate corporate policy and standards. This communication is the responsibility of the project manager or team leader. If there was a failure to communicate specific policy on the part of the project manager, you are not liable for the security breaches incurred because of it.

SUMMARY

Determining responsibility due to a failure to follow corporate policy is a common problem. In most cases, the programmer will take the heat simply because they are lower than management on the totem pole. Failing to implement security standards is a bit more serious because of the implications to the business and system integrity. There are some areas where programmers need to be more proactive; this may be one of them.

Temporary Fixes – A Common Practice

A temporary fix is usually a workaround that makes a computer program function correctly but is not the optimum solution for the problem. The next issue addresses the weaknesses and responsibility for implementing and updating a temporary fix.

You are past your deadline and have a couple of fixes that must be implemented in order for your code to run properly. Time is very tight so you apply a temporary fix with the intention of creating a more secure fix when you have more time to spend on it. Is it ever appropriate to implement an insecure temporary fix when the pressure is on for project completion?

Conservative It is never appropriate to implement a fix on a computer program, even if it is temporary. A temporary fix can result in a breach of system security. In the above circumstance, you should simply tell your manager that the appropriate fix will take more time and consequently you will not meet your deadline. If your manager tells you to apply the quick fix explain to them that it will not work.

Liberal In a perfect world, it is always better to take the time to create a secure program fix. The reality of computer programming paints a vastly different picture then the ideal world. Sometimes there are aggressive deadlines that must be met, and it is common practice to implement patches after a software release.

SUMMARY

This issue weighs the business realities against the technological demands and the results of those high-pressured demands. The reality concerning temporary fixes is that they are common practice. These fixes are systematically followed up with software updates and patches. Some of the most well known software companies in the world such as Microsoft make this common practice in the case of operating system patches that address security weakness.

Disabling Warnings in Macros – Not a Big Deal?

You are an Excel programmer and are annoyed with the system window pop-up every time you open your Excel code, which says that macro viruses are possible and asks are you sure you want to open this application. You decide to disable the warning. Is it ethical to disable system warnings for your own expedience and convenience?

Conservative It is always better to stay on the safe side and not disable any system pop-up windows. They are there for a purpose and even if it slows you down a bit, do not alter or remove them for your own convenience.

Liberal As an Excel macro programmer, you have expertise in this area and clearly know what you are doing. Disabling a pop-up window that is simply annoying is not a big deal.

SUMMARY

Programmers adjust their systems on a regular basis to make their jobs easier. Since they have more knowledge than the average user of the operating systems they are conducting development efforts in, they may know how to streamline simple system warnings to make their jobs easier and more efficient. It is important to remember that sometimes a little knowledge can be a very dangerous thing. If you decide to circumvent system pop-ups or other system procedures, be certain you know exactly what you are doing.

Using Back Doors – A Catch 22?

Back doors are created by application developers to provide a way into the system in case something goes wrong with the primary application. For example, if the application locks up you will need a way to get in and free the system resources. Without a back door, you cannot do this.

Knowing that it is a security risk to use back doors in your application development, do you do it anyway because you know you will need to get into the system via other means if a problem occurs? Is the creation of a back door appropriate in this case?

Conservative Never put the system you are developing in at risk by using back doors. It is as simple as that. If something does go wrong in your application, it will be difficult for you to get in but not worth the security risk you would otherwise impose on the system.

Liberal This issue is a catch 22 scenario. The security policy rightly states that you should not use back doors when developing applications and this makes complete sense. Back doors are a huge security risk. Nevertheless, if there is a problem with the system, you will be required to quickly get into it by some means to prevent major consequences, which requires you to create a back door. As the programmer, you have to have a back door so you create one knowing the risk.

SUMMARY

Back doors to applications are a significant security risk, but not being able to get into an application is a serious development issue when a problem arises. Speaking to your system administrator may give you new ideas for handling this catch 22 situation. They may have alternate means for handling this problem besides the use of a back door.

Cracking Passwords – Should You or Shouldn't You?

Cracking passwords occurs for malicious reasons as well as innocent ones. The next issue discusses one such potentially innocent reason to crack the system administrator's password if you are a programmer.

You need to install your software onto the test box and the system administrator with the password is out sick. You can guess or crack the password of the test box and therefore install your software. Is it ethical to do this?

Conservative You should wait until the system administrator returns to work. You have not been given blanket access to the test box and therefore you should certainly not try to crack the administrator's passwords. If something goes wrong on that server, you will be the one held responsible.

Liberal Since you are under a tight deadline and can guess the system administrator's password, go ahead and get started on your work.

SUMMARY

If you cannot wait another day because you are under such a tight deadline, reference the employee address book and call the system administrator at home for the password and permission to use the testing box. In most cases, they will not give you the password on the phone but may be able to provide you with the appropriate privileges remotely so that you can use the test machine under your own user identification.

Software Deployment – What is the Best Way?

Deploying software is the final step in the development process not including ongoing maintenance. Within the area of deployment, the developer faces additional ethical concerns. One such concern is departmentalized deployment.

Departmentalized deployment is a means to deploy a recently developed software application that isolates different departments, rolling out the software one department at a time. This allows the development team to see how the new software affects each department and their corresponding systems. This process simplifies and provides easier isolation of problems and fixes.

You know it is more secure to deploy software in a departmentalized manner; however, you do not have the time so you roll it out all at once. Are you wrong in doing this?

Conservative Deployment of software is a very important step in user acceptance. If it is deployed in a rushed manner, it could potentially cause an inconvenience for the users. They will not want to use the new system in the future. This will stain the image of the new application. It is better to take the time and deploy the new application in a departmentally structured manner.

Liberal Again, we are faced with the reality of the software development process against the ideal circumstances of development. Sometimes you do not have the time because the system you built is needed immediately so you must deploy and hope there are not any major problems. In addition, global deployment can be somewhat protected by testing in a simulated test environment that mirrors the development environment.

Summary

The results of deploying applications all at once can pose a problem with regard to user acceptance for that application. However, if all goes well, it is a non-issue. Only the developers and the testing team can make this call.

Chapter Summary

In this chapter we discussed the ethical considerations of programmers and analysts. We defined programmers and analysts as the information technologists who develop computer applications utilizing various programming languages. You learned that there are many different titles a programmer can have, which identify whether they develop package software solutions, operating system functional programs, or other forms of application development.

In the conservative and liberal ethical responses, we reviewed bad and weak coding practices. We included the ethics of code reviews comparing automated reviews with peer and third-party reviews. You learned about the relationship between programmers and malicious attacks such as viruses and stealing. You learned how to ethically handle these circumstances. In addition, we considered the ethical responsibility of keeping information systems secure when writing code and the expected responsibility of the developer. Finally, we discussed the ethics of application deployment, structured design, and testing.

This chapter thoroughly highlighted the ethical pitfalls of computer programming, making you aware of what risks you could consider taking to speed up production and what risks are simply not worth the results.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to get you thinking about the ethical circumstances you may face when performing computer programming or systems analysis. Unless legal issues are involved, then answers in the FAQ may not be the right answer for your organization. Some answers may be more ethical than others, but the true response is up to you.

Q: What is bad code and what are the ethical ramifications of writing bad code?

A: Bad code is code written that does not fully take into account the technical specifications provided and falls short of the application requirements. In some cases, people who are not qualified for the work at hand write bad code. Ethically speaking, if a corporation hires you to do a job, you should have the appropriate capability to perform that function in a time-efficient manner. If you take a job that you are not qualified for, you can tarnish your reputation as a programmer and influence the productivity of your department.

Q: Describe the code review process including potential ethical pitfalls within these processes.

A: The code review process is the phase in the software development life cycle where checking of code occurs for security weakness and overall usability. Ethical issues occur in the code review process when this step of the application life cycle development is skipped or glazed over.

Q: What are the ethical responsibilities of computer programmers when it comes to the generation of viruses due to their negligence?

A: Computer programmers have the unspoken responsibility to prevent virus and worm infection on company resources. When programmers do not handle their equipment in a diligent manner, they put the business at risk of contamination. This carelessness can cause serious damage and impedes the development process altogether.

Q: Do programmers carry the weight of security or is this just the role of the Information Security Officer?

A: All personnel within a corporation should be security conscious. This is especially true of computer programmers. When developers implement insecure code or back doors, they put the company at risk. It is up to the programmers to calculate if the risk is necessary given the application requirements.

Q: In application deployment is there an ethical requirement of the developer to perform deployment in a certain manner?

A: The answer to this question is heavily dependent upon the expected project deadlines and physical deployment environment. In some cases, global deployment is necessary and can be somewhat protected by testing in a simulated test environment that mirrors the development environment. In other circumstances, avoiding departmental deployment out of laziness can be disastrous.

Q: As a developer, are you required to follow a code design process, or is it acceptable to handle coding errors in the testing phase of application development?

A: Again, this question depends on the environment of the IT department and their expectations. The code design process does circumvent future reworking of the code at the testing phase. However, your technical lead may not require or desire you to perform a code design phase within the development life cycle.