CHAPTER 12

# Software Support and Maintenance

## OBJECTIVES

- Describe postrelease product support and customer service activities.
- Discuss product defect and nondefect support tasks.
- Analyze a sample customer support and service organization.
- Describe product-fix releases, which contain only product defect fixes, and maintenance releases, which contain small product enhancements and defect fixes.

## 12.1 Customer Support

Large software products go through a very lengthy and expensive development process. However, the postrelease product support and maintenance cycle is several times longer than the original development cycle. The software product life span after the initial release lasts many years. The success of the product is dependent on the real users' experience with the software. Despite developers' hopes to have "user friendly" and "high-quality" software, many large software products are complex and contain defects at product release time. Furthermore, due to a variety of reasons such as resource and schedule constraints, the first release of a product may not include everything in the requirements list. Thus it is extremely important that there is a good understanding of and extensive preparation made for postrelease customer support. The customer support and services include two main types of activities:

- Product defect support
- Product nondefect support and services

In addition, most of the customer support and service for large software products charges a fee. Many software organizations charge as much as 10% to 20% of the original product price for one year of postrelease product support and customer service. Some charge a fee on a "per-call" basis where each telephone call to the customer support service requires the caller to pay for the support/service effort. These paying customers and users clearly expect a professional level of services and product maintenance updates. A number of the product support and customer service models extend the support and service into a customer consulting practice. The consulting service will include customer product education and mentoring as well as performing product extensions and customizations. In general, once a software product is modified and customized, that portion is not covered by the general product support contract and must be supported by the consulting service contract. We will not discuss the extended model of consulting services in this text except to mention that it is a growing but difficult business. Companies such as IBM and Wipro of India have made IT consulting a global business.

For those using or planning to use open source software, product support and customer service may be a problem. Because the software product is free, there is no legal obligation to support it. For some open source software, there may be a commercial support provider, but usually it is for a popular product that is in demand. For example, Red Hat is a company that provides support for Enterprise Linux, which is a free operating system, and for JBoss Enterprise Application Platform.

### 12.1.1 User Problem Arrival Rate

Once a software produce is released, there is usually a surge of question and problem reports from users as the early usage rate picks up. It is well known that the problem discoveries and reports arrivals follow an exponential curve much like the Rayleigh curve shown in **Figure 12.1**. However, the real data are not as smooth as what is depicted in **Figure 12.1**.

The problem arrival rate increases very quickly in the beginning as users try a new software product. In the beginning, the users are exercising the main paths and finding all the easy "low-hanging fruit" problems. As these major problems get fixed and user

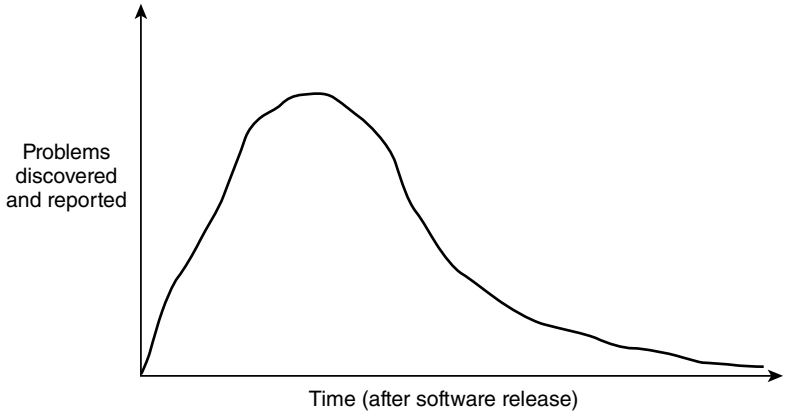


**Figure 12.1** A Rayleigh curve illustrating problem arrival.

sophistication increases, the problem reporting rate will also decrease to a lower rate. Sometimes a software product may be mistakenly thought to be of high quality when the initial problem reports come in very slowly. When this happens, it is important to check the actual usage rate of the product because many customers may not immediately use newly released software. Some software engineers assign the term *usage-time* to the time axis in **Figure 12.1** and thus expand strict calendar time to depict the problem arrival curve. A usage-time unit is often defined as follows:

> Usage months = (Number of users actively using the software) X (Number of months of usage)

This formula ensures that both time and usage are included when depicting the problem arrival rate. Kan (2003) contains an extensive discussion on the Rayleigh model and defect arrival rates. These defect arrival rates also resemble the test defect discovery rates and reliability models during testing. (See Musa, Iannino, and Okumoto [1990] for additional details on reliability models.)

The preparation for customer support must start before the software is released. The support and service workload and the number of support personnel required are estimated from the problem arrival curve. Therefore, it is vital that there is some way to accurately estimate the problem arrival rate. For those organizations having a history of usage data, the estimation of problem arrival rate of a similar situation is quite feasible. The difficulty occurs when a new product to the organization is being released. The subject of estimation and reliability modeling is beyond the intent of this text. (See the References and Suggested Readings section for references.) Estimating the number of required support personnel is only a first step. The support personnel need to be educated on the software product. Often this is accomplished through education in both the software and the usage environment. Sometimes the support personnel are included in the testing effort as part of the education and transition activities. Other times an original developer or a tester is seeded in the support team to mentor the support team during the first six months or so of the product support period.

In experienced support organizations, detailed accounts of usage and problem reports are constantly and carefully analyzed. The data from the first several months are sometimes used as a checkpoint to see if defect discovery and arrival projections are tracking to projection. During the early support period, a renegotiation of support resources may be needed, especially if more problems are coming in than were originally projected.

The actual cost of maintenance and subsequent releases would depend on the marketing and financial situation of each software product. Those successful software products that have large user communities tend to stay in the market for years with multiple updates and releases. Even then, eventually, the product will enter a sunset period and be taken off the market. The product sunset period may take a year or two and usually go through the following sequence:

1. Stop any product's new feature or functional enhancement release.
2. Fix only very high-severity problems related to the old product.
3. Announce a new replacement product.
4. Encourage existing users and customers to migrate to the new product.
5. Notify users remaining on the old product of the planned termination of product support.
6. Provide customers who are still remaining on the old product with names of possible software vendors willing to continue the old product support.
7. Terminate customer support on the planned date and withdraw the software product from the market.

Sometimes, a software product is withdrawn, but there is no new replacement product. In such situations, the sunset period may be a lot longer in order to allow users to find an alternative solution.

### 12.1.2 Customer Interface and Call Management

A typical customer support organization has several layers of personnel performing different types of activities. **Figure 12.2** is an example of a software support organization with two layers of support and service functions. There is an outer layer handling the interfaces with the customers, and there is an inner layer of technical experts diagnosing and fixing the more difficult problems, including changing the code.

The outer layer is composed of the service and support representatives who represent the software product and the service functions to customers and users. At the same time, these service and support representatives are advocates for the customers and represent the customers' view of how the product needs to be fixed or modified. These representatives must possess exceptionally good communication skills and application domain knowledge. The inner layer is composed of more technical and product-knowledgeable personnel whose main task is to develop code fixes of the customers' problems. These two groups work very closely and, most of the time, jointly maintain a frequently asked questions (FAQ) database. This database contains a history of past, known problems and the associated fixes, and it may also contain work-arounds for problems that do not yet have fixes. There are also frequent communications concerning customer problems and fix status between the two groups.

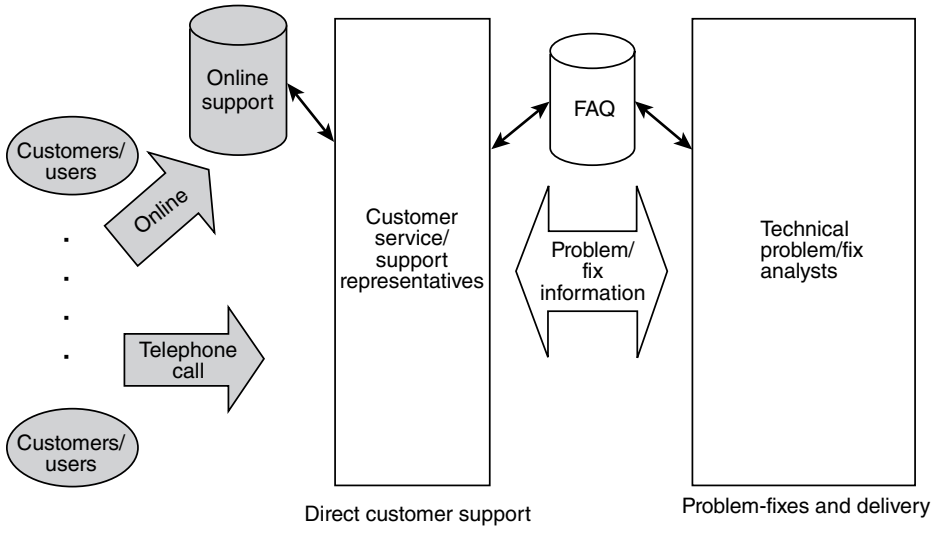


**Figure 12.2** A typical customer service and support organization.

The customer support/service representatives handle customer problems in two major forms:

- Customer telephone calls
- Online submissions

When a customer or user encounters a problem, the traditional mode of operation is to call the customer support representatives. Depending on the size of the support/service organization, there may be an automatic call management tool that directs the incoming calls to different service representatives, based on the type of problem and the availability of the representatives. The support/service representative who receives a call follows a sequence of activities such as the following:

1. Take the customer's name and identification to ensure that this is a qualified customer (e.g., a customer who has paid for the software service).
2. Listen to and record the description of the problem.
3. Perform a scan of the FAQ database to check for similar problems and solutions.
4. Provide the answer if it is a previously reported and already resolved problem.
5. Provide an expected fix date if it is a previously reported but unresolved problem.
6. If it is a new problem, provide a work-around if possible. If there is no work-around, agree on a priority level for the problem with the customer. Based on this agreed on priority level, provide an estimated fix time to the customer.
7. Record the problem and issue a report to the technical fix team.

Call management is the process of dealing with customer calls and ensuring that they are handled smoothly. The customer support/service representatives are encouraged to resolve only those problems that are short and easy to handle over the phone and do not

require code fixes. The automated call management tool also aids supervisors in collecting data on the call traffic, call queue length, call response and process time, number of missed calls, and other relevant information. The number of representatives that are required to support the product and to service the customers depends on the expected call volumes, which in turn depend on the height and shape of the expected problem arrival curve shown in **Figure 12.1**. At times, customers may not be satisfied with a particular service representative's attitude or that person's answer. In such cases, there must be a backup process to handle the unsatisfied customers. In the past, many of these services have been outsourced to off-shore countries, however, companies such as Dell have moved them back close to the corporate home office to improve the support services.

In addition to direct and synchronous call support, support/service representatives are also asked to maintain an online website for asynchronous customer support. Many of the telephone call activities are automated with this asynchronous interface. The support/service website begins by asking for the customer's name and identification and automatically checks the qualification of the customer. The customer's reported problem is automatically checked against the FAQ database and a response is automatically generated, based on whether there is a match. If it is a new problem, the request is automatically reported and later viewed by a customer support/service representative. A response is then communicated to the customer, along with the representative's assessed problem priority level and estimated fix date. This response may be a direct email or a posting that the customer may check later, or both. There may also be a customer escalation facility, which handles dissatisfied customers. Some support organizations will post some product marketing announcements and future release announcements on the support/service website. Others will post versions of the actual code fixes on the site, allowing the customers to directly download the fixes and install them themselves.

All experienced customers know that the way to receive fast and quick consideration is to have the problem priority assessed as high as possible. They are keenly aware of the relationship between the problem priority level and the problem-fix response time. **Table 12.1** gives an example of this relationship. Many experienced customers will ask for this type of service agreement before signing up and paying for the customer support/service contract. A particular concern will be finding out if the customer call center operates 24 hours a day for 7 days a week. Today, most customers expect this level of responsiveness.

### 12.1.3 Technical Problem/Fix

If a problem is beyond the ability of the support/service representative, then the technical problem/fix analyst is brought in. There may be a work-around or a temporary fix that

**Table 12.1** SampleProblem Priority Levels

| Priority Level | Problem Category | Fix Response Time |
|---|---|---|
| 1 | Severe functional problem with no work-around | As soon as possible |
| 2 | Severe functional problem but has a work-around | 1-2 weeks |
| 3 | Functional problem that has a work-around | 3-4 weeks |
| 4 | Nice to have or to change | Next product release or earlier |

may be passed to the customer. In that case, both the problem description and temporary solution is logged in the FAQ database to help other customers who encounter a similar problem. Meanwhile, a formal change request is created so that a more permanent fix, which may involve a code change, is generated. Based on the change request, the solution to the problem is designed, coded, and tested. A very high priority fix will be packaged and made available to customers immediately. It will also be integrated into a fix release, which is a collection of problem-fixes that is released periodically. The non-high-priority fixes are all integrated and packaged into periodic fix releases.

The technical problem/fix analysts are usually well-qualified engineers who are experts in designing coding, and testing. However, different from development software engineers, these problem/fix analysts have neither the support of the product requirements analysts nor the original product designers. Thus they have to possess both the product-specific, industry knowledge as well as general technical knowledge. While this is a management problem, the appreciation and recognition of these technical problem/fix analysts has been long overdue. Their special value is just starting to be recognized.

The process of passing a customer problem from the support/service representative to the technical problem/fix analyst and getting the solution back to the customer is almost the same as the one used between testers and developers. The following is a high-level summary of such a process:

1. Problem description, problem priority, and other related information is recorded in a problem report that is submitted to the problem-fix-and-delivery group.
2. The problem-fix-and-delivery group will explore and analyze the problem, including the reproduction of the problem.
3. The problem-fix-and-delivery group either accepts or rejects the problem.
4. If the problem is rejected, the direct customer support group is immediately notified. If the problem is accepted, then a change request is generated and the problem enters a fix cycle of design, code, and test.
5. Depending on the priority and nature of the problem, the fix may be individually packaged and released immediately to the customer or the fix may be integrated into a fix release package.
6. The FAQ database is updated to reflect the status of all the problems so that the customer support/service representatives may quickly and accurately advise the customers on the problem resolution status.

The design, code, and test cycle of a problem-fix is not very different from the design, code, and test activities of the development cycle. The danger is that the fix cycle may not utilize the same level of software engineering discipline because many of the problem-fixes may involve only one or two lines-of-code fix. It is so tempting to just fix the code and go on to the next problem. For an organization that is handling multiple software product releases across multiple countries, as mentioned in Chapter 11, the discipline of keeping all the requirements, design, coding, and testing documentations updated is imperative. A code fix to a problem often needs to be propagated to other releases and also to multiple country versions. The tasks and activities of the problem-fix-and-delivery group require the same software engineering methodologies and techniques as those described in the earlier chapters of this text. In addition, there is more awareness of the need for measurements as

customers are starting to demand contractual agreements on service levels with penalty clauses if the agreement is not met. This is especially true with hardware system availability where most of the system is expected to be available between 99% and 100%. Fortunately, the customer demand on software has not yet reached such a level.

The problem/fix process for life-threatening situations is very different from the normal condition. It is not uncommon to see support personnel physically on site at a customer's system to fix and resolve the problem in such emergency situations. In fact, depending on the particular case, even the original designer or programmer sometimes participates in supporting the customer. In such situations, the changes and fixes made at the customer site still need to be documented and the fix code integrated back into a general fix release later for other customers. A word of caution: Support is only for problems originating from the original software product, not for any modified or customized code problem. Today, most software product organizations provide only executable object code and do not release source code anymore to evade the issue of supporting customer-modified code.

### 12.1.4 Fix Delivery and Fix Installs

The deliveries of fixes follow two major paths. There are the regular and periodic fix releases that are on a quarterly or semiannual cycle. These are the different code fixes that have been accumulated since the last fix release. A fix release, in general, does not contain the complete product. It is not a total product refresh but contains only those modules and code that are affected by the problem-fixes. The fix code is packaged in a sequential order. The fix releases are also sequential, and the customers are expected to install the fix releases in sequential order. All fix releases are made available to all the customers that have a valid support/service contract. These fix releases, sometimes called patches, are downloadable from an online site or are shipped to the customers on CDs.

Not all customers will apply (or install) the fix releases immediately. These are customers who have achieved a "working" state and do not want to risk their working software with any new fix releases. They may, however, encounter a problem later and want a particular problem-fix. These customers must be reminded to apply all the fix releases in sequence since they last fully applied a fix release. Such a retroactive application of all the fix releases can be very time consuming and frustrating when the customer wants only one problem fixed or a single patch. However, in order to keep the support service effort at a reasonable level, customers are always encouraged to apply the fix release as soon as possible after it becomes available.

The other path of fix delivery is for the high-priority emergency fixes. In such a situation, a single fix may be delivered to the customer whose system may be down and whose business is suspended. The customer will immediately apply the fix upon receiving it. The customer's system is brought up and business resumes. However, those customers who applied this emergency fix must be cautioned when the next regular fix release is made available. They should not perform a partial install of the fixes in the fix release, figuring that they already have the emergency fix applied. It is possible that some of the fixes in the accumulated fix release may impact the emergency fix that was already applied. At times, the customers who have applied the emergency fix may have to back out of that emergency fix first and then apply the complete fix release. **Figure 12.3** depicts a fix release (n+1) that contains three accumulated fixes.

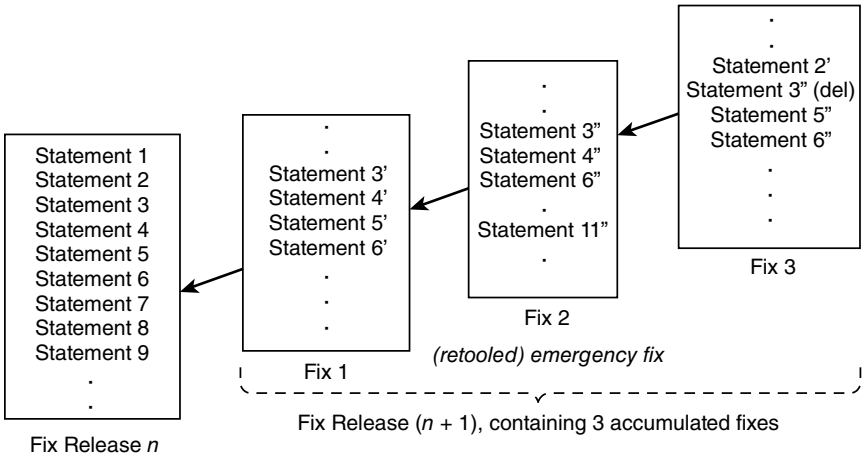


**Figure 12.3** Fix overlay problem.

Suppose there is an emergency problem and an emergency fix is delivered to those customers who are at `Fix Release n` level in **Figure 12.3**. Assume that the emergency fix sent to the customer at `Fix Release n` level touches code statements `3`, `4`, and `5`. Meanwhile there is already a new, next code fix, `Fix 1`, for problem 1 that is placed into the integration bucket for the next fix release, `Fix Release n+1`. Furthermore, suppose that because of the `Fix 1`, which affected the `Fix Release n` (`Statements 3`, `4`, `5`, and `6`), the already delivered emergency code fix must now be altered to `Fix 2` because it touches the same area. Then any subsequent problem-fix, `Fix 3`, must be based on this altered emergency fix, `Fix 2`. This is an example of why an emergency fix cannot be left in the code and why customers cannot just blindly apply the fix release.

The delivery group of the customer support/service organization must keep track of the delivery status and customers' fix apply status. When a new fix release is available, an information document that contains the list of problems and corresponding fixes is included along with advice to customers who may be at different release, version, or fix release levels. In addition, there are times when software applications may be so intertwined that a fix release of one must be coapplied with a fix release of another software product. For example, an operating system fix release may be related to a fix release of a database system and of a network system. These three fix releases are corequisites of each other and must be applied together for all the fixes to work. This corequisite situation is discussed more in the next section and is shown in **Figure 12.4**. The various fix release information is also shared with the customer service support representatives so that when a customer reports a problem, it is clear what level the customer is at with which product.

## 12.2 Product Maintenance Updates and Release Cycles

In the previous sections, we discussed the customer support/service activities and the concerns of fix releases and applying emergency fixes. Aside from product defect problems, there also are small, functional enhancements provided to customers as maintenance updates.

Usually, when there is a major functional enhancement or a new platform supported, the software product will be newly priced as a different product release. These new product releases are also developed by the product development organization as opposed to the customer support/service organization. However, these small, incremental changes are not treated as a new product release. They are often included as a part of the customer support/service contract. An example is the addition of the support of a new device, such as a new printer. Another example, for a payroll application, may be the changes needed to accommodate the annual tax law changes. These product updates are usually integrated with the product defect fixes release and delivered as a product maintenance update.

Different software application products will require different product maintenance cycles. Some applications will go through a maintenance update every six months, and others will have an annual update to match legislative updates. **Figure 12.4** illustrates three separate but related commercial software applications that are releasing their updates at slightly different periods. Each product organization can have their separate release plans, except when there are product linkages. For example, if a new feature modification in the manufacturing application is related to some area in the distribution application, then their respective update/fix maintenance releases must match and be made available simultaneously. The Update/Fix Maintenance Release 2 of Manufacturing may be related to the Distribution application's Update/Fix Maintenance Release 1. Then these two maintenance releases must be timed to come out at the same time, and the updates/fix maintenance releases for the two software products must list each other as corequisites in their respective documentations, which are shipped along with the code. The customers must apply

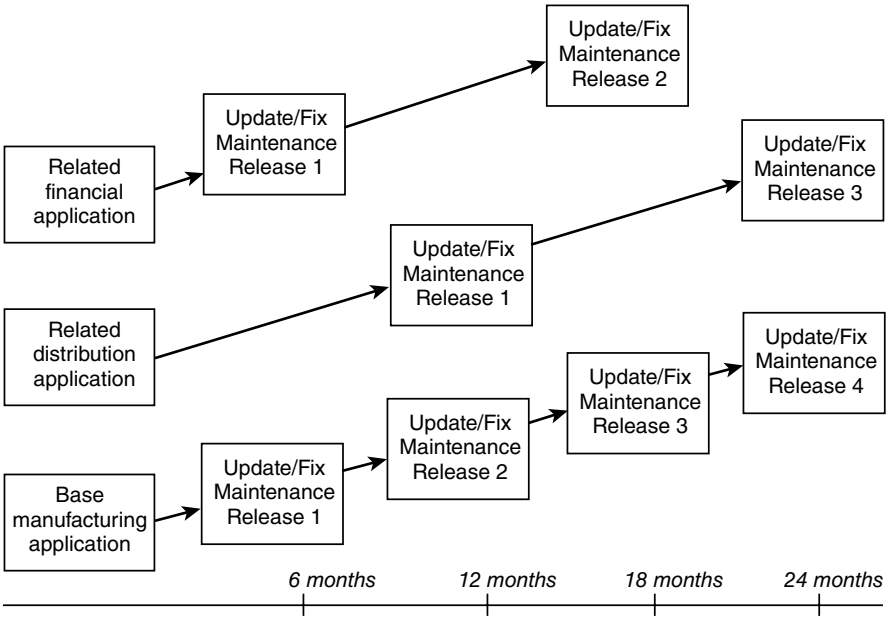


**Figure 12.4** Multiple-product fix releases.

both of these at the same time. Otherwise the new feature may not work. Cross software applications planning is a complex and vital set of activities that involves both technical and business knowledge. Its significance is gaining more attention as different software companies strike up business alliances and partnerships, not to mention corporate mergers and acquisitions. It is also clear that configuration management, discussed in Chapter 11, and an automated configuration management tool are necessities in order for customer support and service to succeed.

## 12.3 Change Control

In addition to the need for configuration management, a large customer support/service organization must have a well-established change control. Just as development organizations must have good change control or risk scope creep and blown schedules, the customer support and service organization must control all the changes, whether the change originated from a customer problem or a planned small, incremental, feature enhancement. The scheduling of changes based on the number of anticipated problems and change requests is usually resource gated. There may be special situations when an unexpectedly large volume of change requests can drive a customer support/service organization to increase their staffing with temporary help personnel. Due to the dynamic nature of many of the customer support/service organizations, it is even more crucial to have a good change control process.

Change control is a process that manages the flow of the following activities:

- Origination of a change request
- Approval of a change request
- Acting on a change request
- Tracking and closing a change request

Along with the activity flow, there is also certain information that must be passed through the flow. The information relevant to a change request is usually captured in a change request form. **Figure 12.5** shows a sample of such a form.

For easy identification and tracking purposes, each change request is given a request number. The date of the initial request and the requestor's name are also needed, and the change request should include a priority indication of high, medium, or low. The priority level provides the information for the decision process of which maintenance release should the solution to the change request be placed in. Thus it serves not only as a priority for the work order but also as a priority for making the solution available to customers. The request status allows the tracking and managing of the change request. Note that not every change request is approved. Some are rejected. Although not included in **Figure 12.5**, a more sophisticated form would include a field for a rejection reason code. The request may be in the process of being worked on, and the process start date would be used for tracking. When the change request is fulfilled, the completion date is filled in. There is an area for a brief description of the change request. If more space is needed, an attachment may accompany the request form. In order to ensure sufficient analysis has been performed, the request impact analysis field is included. The impact analysis should include the names of all the modules, database tables, and so on, that are affected. The impact analysis makes it

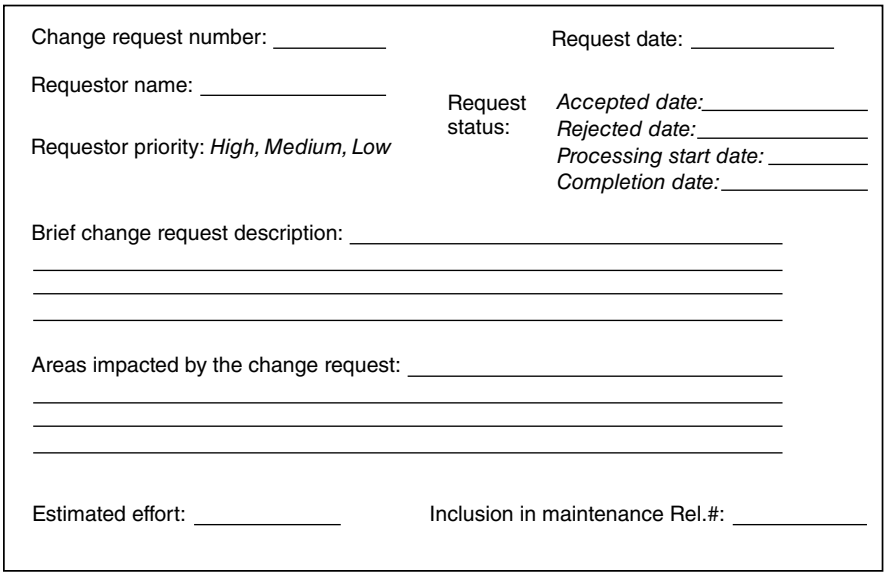


**Figure 12.5** Sample maintenancy change request form.

possible to determine all the corequisite and prerequisite changes that must accompany this change request. The tighter the coupling among components, the more prerequisite and corequisite relationships will exist. The decisions made at the design stage may come back and adversely affect the product support and maintenance cycle. Work effort estimation is also needed for the change request to be evaluated for planning the required resources and scheduling a potential completion date. In addition, there needs to be an estimate of which upcoming release the change request would be included in.

There are more comprehensive change request forms for organizations that desire to manage the change request more thoroughly. For example, the estimated effort field may be modified into two fields that capture the initial effort estimate and the actual effort expended. Also, there may be a field that shows the actual modules, database, and so on, that were impacted by the change request. However, it is important that the form is not made more complex than necessary. There must be a good reason to have a field included in the form. The information such as the brief change request description and impact areas, besides serving as information for accepting or rejecting the request, are all rolled into a customer support newsletter that either precedes or is sent together with the maintenance/fix release code.

Many software organizations perform extensive studies over these change request forms to better understand their product shortcomings, customer needs, and future product directions. An experienced customer support and services group can greatly enhance the customer's perception and sales of the product. Those who have extended the support and services group into a consulting service have also seen some outstanding revenue opportunities. The customer support/service organization performs a wide variety

of activities related to both product defect and nondefect support. Some even perform marketing-related activities such as customer opinion surveys.

## 12.4 Summary

This chapter describes both defect and nondefect support and services. The support activities start as soon as a software product is released. The customer problem reports that come into the support and service organization are found to follow an exponential track that resembles a Rayleigh curve. The workload and resource effort estimation is often based on the accuracy of this problem arrival curve.

A customer support/service organization is usually composed of two major groups:

- Customer service/support representatives who readily interact with the customers either through telephone calls or online websites.
- Technical problem-fix analysts who work on deeper problem-fixes, including design, code, test, and release.

Various problems associated with applying code fixes and releases are described—from emergency problems to regular and periodic maintenance releases.

Finally, all maintenance changes must be controlled with change requests as done in the product development cycle. A maintenance change request form is discussed in terms of its usage in controlling change requests.

## 12.5 Review Questions

1. List three customer support functions that a customer support/service organization performs.
2. Explain the customer problem arrival curve in terms of customer usage of the product and the fixes.
3. What is the formula for usage month?
4. What is a prerequisite/corequisite relationship of product maintenance and fix releases?
5. What is a problem priority level? What is it used for?
6. Describe the steps involved when a customer problem is passed from the customer service/support representative to the technical problem/fix analyst until the problem is resolved.
7. Give an example of a problem that may occur if a customer stays on a particular release, skips several maintenance/fix releases, and then applies a fix release.
8. What is the estimated effort field on the change request form used for?

## 12.6 Exercises

1. Visit the online customer support site of a software product company and compare the list of support functions it offers against your list from Review Question 1.

2. Explain what the problem arrival curve may look like if a software product that is released does not get installed and used until six months later.
3. In addition to prioritizing the work order itself, what is the change request priority information on the change request form used for?
4. Consider a situation where an important customer has installed a generic software application and also customized some parts of the application. Explain the effect that would have on supporting this customer if under the following circumstances:
   a. The customized code is only in interfacing with an application that the customer wrote in-house.
   b. The customization is only in adding an entry in a database table.
   c. The customized code is in the main logic of the purchased software application.

## 12.7 References and Suggested Readings

V. Basili, et al., "Understanding and Predicting the Process of Software Maintenance Releases," *Proceedings of the 18th International Conference on Software Engineering*, March 1996.

S. Dart, A. M. Christie, and A. W. Brown, "A Case Study in Software Maintenance," Software Engineering Institute, CMU/SEI-93-TR-8, Carnegie Mellon University, 1993.

S. H. Kan, *Metrics and Models in Software Quality Engineering*, 2nd ed. (Reading, MA: Addison-Wesley, 2003).

T. Kilpi, "New Challenges for Version Control and Configuration Management: A Framework and Evaluation." In *Software Engineering: Selected Readings*, edited by E. J. Braude (Piscataway, NJ: IEEE, 2000): 307–315.

J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability* (New York: McGraw Hill, 1990).

Red Hat, http://www.redhat.com, accessed 2016.

G. Ruhe and M. O. Saliu, "The Art and Science of Software Release Planning," *IEEE Software* (November/December 2005): 47–53.

Software Support and Maintenance FAQ, http://faq.gbdirect.co.uk/support/, accessed 2016.