# Software Metrics and Problem-Solving

# 10

**Abstract**

This chapter is concerned with metrics and problem-solving, and this includes a discussion of the Balanced Scorecard which assists in identifying appropriate metrics for the organization. The Goal, Question, Metrics (GQM) approach is discussed, and this allows metrics related to the organization goals to be defined. A selection of sample metrics for an organization is presented, and problem-solving tools such as fishbone diagrams, Pareto charts, trend charts are discussed.

**Keywords**

Measurement · Goal, Question, Metric · Balanced scorecard · Problem-solving · Data gathering · Fishbone diagram · Histogram · Pareto chart · Trend graph · Scatter graph · Statistical process control

## 10.1 Introduction

Measurement is an essential part of mathematics and the physical sciences, and it has been successfully applied to the software engineering field. The purpose of a measurement programme is to establish and use quantitative measurements to manage the software development processes and software quality in an organization; to assist the organization in understanding its current software engineering capability; and to provide an objective indication that software process improvements have been successful.

Measurements provide visibility into the various functional areas in the organization, and the quantitative data allows trends to be seen over time. The analysis of the measurements allows action plans to be produced for continuous

© Springer International Publishing AG 2017

G. O'Regan, *Concise Guide to Software Engineering*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-57750-0_10

improvement. Measurements may be employed to track the quality, timeliness, cost, schedule and effort of software projects. The terms "*metric*" and "*measurement*" are used interchangeably in this book. The formal definition of measurement given by Fenton [1] is:

> Measurement is the process by which numbers or symbols are assigned to attributes or entities in the real world in such a way as to describe them according to clearly defined rules.

Measurement plays a key role in the physical sciences and everyday life, for example, calculating the distance to the planets and stars; determining the mass of objects; computing the speed of mechanical vehicles; calculating the electric current flowing through a wire; computing the rate of inflation; estimating the unemployment rate. Measurement provides a more precise understanding of the entity under study.

Often several measurements are used to provide a detailed understanding of the entity under study. For example, the cockpit of an airplane contains measurements of altitude, speed, temperature, fuel, latitude, longitude, and various devices essential to modern navigation and flight, and clearly an airline offering to fly passengers using just the altitude measurement would not be taken seriously.

Metrics play a key role in problem-solving, and various problem-solving techniques will be discussed later in this chapter. Measurement data is essential in quantifying how serious a particular problem is, and they provide a precise quantitative measure of the extent of the problem. For example, a telecommunications outage is measured as the elapsed time between the downtime and the subsequent uptime, and the longer the outage lasts the more serious it is. It is essential to minimize outages and their impact, and measurement data is invaluable in proving an objective account of the extent of the problem. Measurement data may be used to perform analysis on the root cause of a particular problem, e.g. of a telecommunications outage, and to verify that the actions taken to correct the problem have been effective.

Metrics provide an internal view of the quality of the software product, but care is needed before deducing the behaviour that a product will exhibit externally from the various internal measurements of the product. A *leading measure* is a software measure that usually precedes the attribute that is under examination; for example, the arrival rate of software problems is a leading indicator of the maintenance effort. Leading measures provide an indication of the likely behaviour of the product in the field and need to be examined closely. A *lagging indicator* is a software measure that is likely to follow the attribute being studied; for example, escaped customer defects are an indicator of the quality and reliability of the software. It is important to learn from lagging indicators even if the data can have little impact on the current project.

## 10.2 The Goal, Question, Metric Paradigm

Many software metrics programmes have failed because they had poorly defined, or non-existent goals and objectives, with the metrics defined unrelated to the achievement of the business goals. The *Goal, Question, Metric* (GQM) paradigm was developed by Victor Basili and others of the University of Maryland [2]. It is a rigorous goal-oriented approach to measurement, in which goals, questions and measurements are closely integrated.

The business goals are first defined, and then questions that relate to the achievement of the goal are identified. For each question, a metric that gives an objective answer to the particular question is defined. The statement of the business goal is precise, and it is related to individuals or groups. The GQM approach is a simple one, and managers and engineers proceed according to the following three stages:

- Set goals specific to needs in terms of purpose, perspective and environment.
- Refine the goals into quantifiable questions.
- Deduce the metrics and data to be collected (and the means for collecting them) to answer the questions.

GQM has been applied to several domains, and so we consider an example from the software field. Consider the goal of determining the effectiveness of a new programming language *L*. There are several valid questions that may be asked at this stage, including who are the programmers that use L?; and what is their level of experience?; What is the quality of software code produced with language L?; and What is the productivity of language L? This leads naturally to the quality and productivity metrics as detailed in Fig. 10.1.

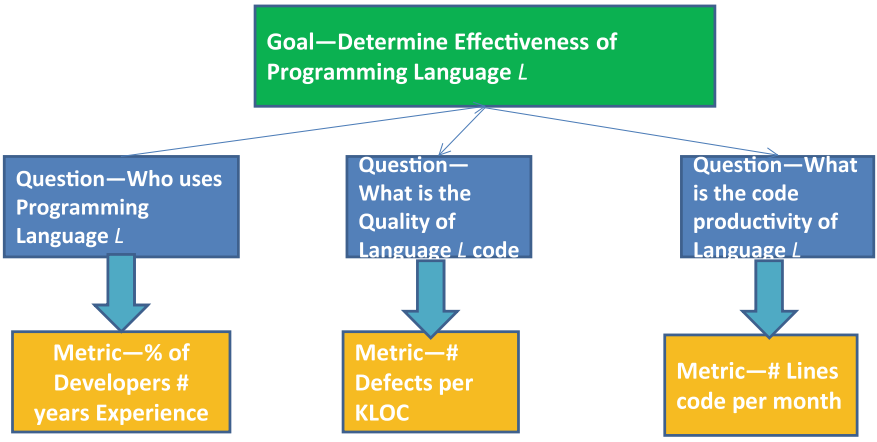


**Fig. 10.1** GQM example

*Goal*

The focus on improvements should be closely related to the business goals, and the first step is to identify the key goals that are essential for business success (or to the success of an improvement programme). The business goals are related to the strategic direction of the organization and the problems that it is currently facing. There is little sense in directing improvement activities to areas that do not require improvement, or for which there is no business need to improve, or from which there will be a minimal return to the organization.

*Question*

These are the key questions that determine the extent to which the goal is being satisfied, and for each business goal the set of pertinent questions need to be identified. The information that is required to determine the current status of the goal is determined, and this naturally leads to the set of questions that must be answered to provide this information. Each question is analysed to determine the best approach to obtain an objective answer, and to define the metrics that are needed, and the data that needs to be gathered to answer the question objectively.

*Metrics*

These are measurements that give a quantitative answer to the particular question, and they are closely related to the achievement of the goals. They provide an objective picture of the extent to which the goal is currently satisfied. Measurement improves the understanding of a specific process or product, and the GQM approach leads to measurements that are closely related to the goal, *rather than measurement for the sake of measurement*.

GQM helps to ensure that the defined measurements will be relevant and used by the organizations to understand its current performance, and to improve and satisfy the business goals more effectively. Successful improvement is impossible without clear improvement goals that are related to the business goals. GQM is a rigorous approach to software measurement, and the measures may be from various viewpoints, e.g. manager viewpoint, project team viewpoint. The idea is always first to identify the goals, and once the goals have been decided, common-sense questions and measurement are employed.

There are two key approaches to software process improvement, i.e. *top-down* or *bottom-up* improvement. Top-down approaches are based on process improvement models and appraisals, e.g. models such as the CMMI, ISO 15504 and ISO 9000, whereas GQM is a bottom-up approach to software process improvement and is focused on improvements related to certain specific goals. The top-down and bottom-up approaches are often combined in practice.

## 10.3 The Balanced Scorecard

The Balanced Scorecard (BSC) (Fig. 10.2) is a management tool that is used to clarify and translate the organization vision and strategy into action. It was developed by Kaplan and Norton [3] and has been applied to many organizations. The European Software Institute (ESI) developed a tailored version of the BSC for the IT sector (the IT Balanced Scorecard).

The BSC assists in selecting appropriate measurements to indicate the success or failure of the organization's strategy. There are four perspectives in the scorecard: *customer, financial, internal process*, and *learning and growth*. Each perspective includes objectives to be accomplished for the strategy to succeed, measures to indicate the extent to which the objectives are being met, targets to be achieved in the perspective and initiatives to achieve the targets. The Balanced Scorecard includes financial and non-financial measures.

The BSC is useful in selecting the key processes that the organization should focus its process improvement efforts on in order to achieve its strategy (Fig. 10.3). Traditional improvement is based on improving quality; reducing costs; and

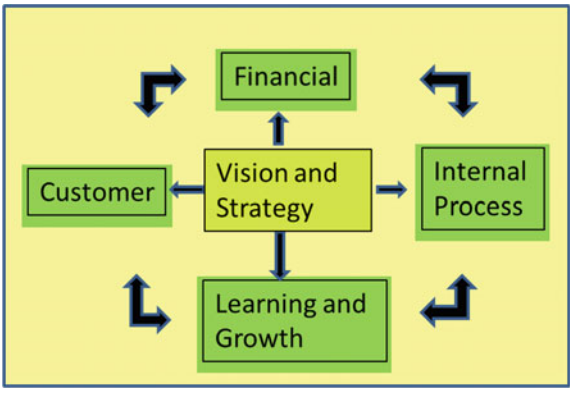


**Fig. 10.2** The Balanced Scorecard

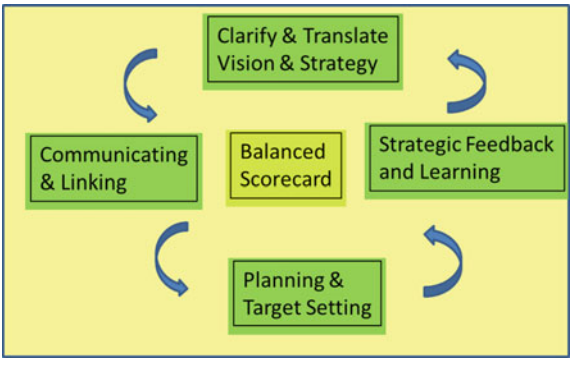


**Fig. 10.3** Balanced Scorecard and implementing strategy

improving productivity, whereas the Balanced Scorecard takes the future needs of the organization into account and identifies the processes that the organization needs to excel at in the future to achieve its strategy. This results in focused process improvement, and the intention is to yield the greatest business benefit from the improvement programme.

The starting point is for the organization to define its *vision* and *strategy* for the future. This often involves strategy meetings with the senior management to clarify the vision and to achieve consensus on the strategic direction for the organization among the senior management team. The vision and strategy are then translated into *objectives* for the organization or business unit. The next step is communication, and the vision and strategy and objectives are communicated to all employees. These critical objectives must be achieved in order for the strategy to succeed, and so all employees (with management support) will need to determine their own local objectives to support the organization strategy. Goals are set and rewards are linked to performance measures.

The financial and customer objectives are first determined from the strategy, and the key business processes to be improved are then identified. These are the key processes that will lead to a breakthrough in performance for customers and shareholders of the company. It may require new processes with retraining of employees on the new processes necessary, and the Balanced Scorecard is very effective in driving organization change. The financial objectives require targets to be set for customer, internal business process and the learning and growth perspective. The learning and growth perspective will examine competencies and capabilities of employees and the level of employee satisfaction. Figure 10.3 describes how the Balanced Scorecard may be used for implementing the organization vision and strategy.

Table 10.1 presents sample objectives and measures for the four perspectives in the BSC for an IT service organization.

**Table 10.1** BSC objectives and measures for IT service organization

| *Financial* | *Customer* |
|---|---|
| Cost of provision of services | Quality service |
| Cost of hardware/software | Reliability of solution |
| Increase revenue | Rapid response time |
| Reduce costs | Accurate information |
| Timeliness of solution | Timeliness of solution |
| 99.999% network availability | 99.999% network availability |
| 24 × 7 customer support | 24 × 7 customer support |
| *Internal business process* | *Learning and growth* |
| Requirements definition | Expertise of staff |
| Software design | Software development capability |
| Implementation | Project management |
| Testing | Customer support |
| Maintenance | Staff development career structure |
| Customer support | Objectives for staff |
| Security/proprietary information | Employee satisfaction |
| Disaster prevention and recovery | Leadership |

## 10.4 Metrics for an Organization

The objective of this section is to present a set of metrics to provide visibility into various areas in the organization and to show how metrics can facilitate improvement. The metrics presented may be applied or tailored to individual organizations, and the objective is to show how metrics may be employed for effective management. Many organizations have monthly quality or operation reviews, in which the presentation of metrics is an important part.

We present sample metrics for the various functional areas in a software development organization, including human resources, customer satisfaction, supplier quality, internal audit, project management, requirements and development, testing and process improvement. These metrics are typically presented at a monthly management review, and performance trends are observed. The main output from a management review is a series of improvement actions.

### 10.4.1 Customer Satisfaction Metrics

Figure 10.4 shows the customer survey arrival rate per customer per month, and it indicates that there is a customer satisfaction process in place in the organization and that the customers are surveyed and the extent to which they are surveyed. It does not provide any information as to whether the customers are satisfied, whether any follow-up activity from the survey is required, or whether the frequency of surveys is sufficient (or excessive) for the organization.

Figure 10.5 gives the customer satisfaction measurements in several categories including quality, the ability of the company to meet the committed dates and to deliver the agreed content, the ease of use of the software, the expertise of the staff and the value for money. Figure 10.5 is interpreted as follows:


**Fig. 10.4** Customer survey arrivals


**Fig. 10.5** Customer satisfaction measurements

8–10 Exceeds expectations,
7 Meets expectations,
5–6 Fair and
0–4 Below expectations.

Another words, a score of 8 for quality indicates that the customers considers the software to be of high quality, and a score of 9 for value for money indicates that the customers considers the solution to be excellent value. It is fundamental that the customer feedback is analysed (with follow-up meetings held with the customer where appropriate). There may be a need to produce an action plan to deal with customer issues, to communicate the plan to the customer and to execute the action plan in a timely manner.

### 10.4.2 Process Improvement Metrics

The objective of process improvement metrics is to provide visibility into the process improvement programme in the organization. Figure 10.6 shows the arrival rate of improvement suggestions from the software community. The chart indicates

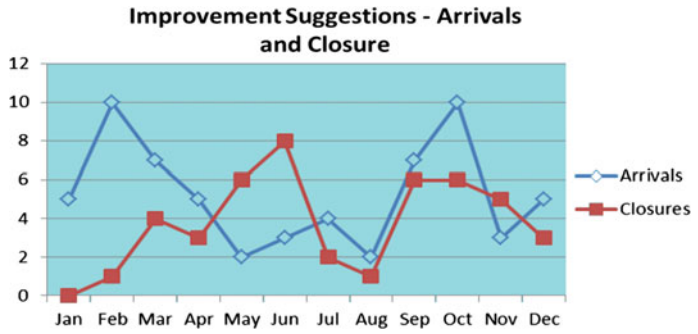


**Fig. 10.6** Process improvement measurements

that initially the arrival rate is high and the closure rate is low, which is consistent with the commencement of a process improvement programme. The closure rate then improves which indicates that the improvement team is active and acting upon the improvement suggestions. The closure rate is low during July and August, which may be explained by the traditional holiday period.

The chart does not indicate the effectiveness of the process improvement suggestions, and the overall impact the particular suggestion has on quality, cycle time or productivity. There are no measurements of the cost of performing improvements, and this is important for a cost–benefit analysis of the benefits of the improvements obtained versus the cost of the improvements.

Figure 10.7 provides visibility into the status of the improvement suggestions, and the number of raised, open and closed suggestions per month. The chart indicates that gradual progress has been made in the improvement programme with a gradual increase in the number of suggestions that are closed.

Figure 10.8 provides visibility into the age of the improvement suggestions, and this is a measure of the productivity of the improvement team and its ability to do its assigned work.

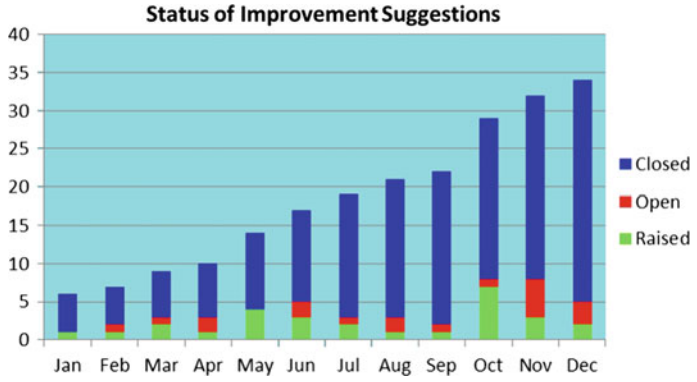


**Fig. 10.7** Status of process improvement suggestions

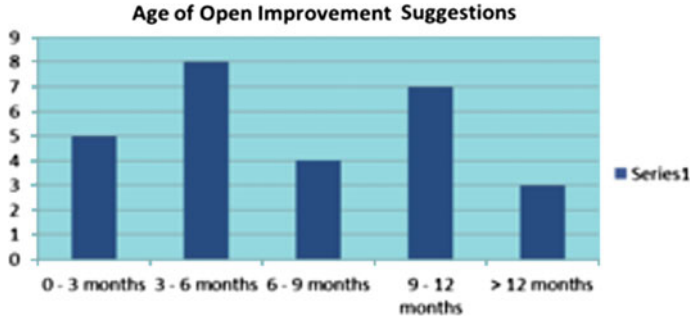


**Fig. 10.8** Age of open process improvement suggestions

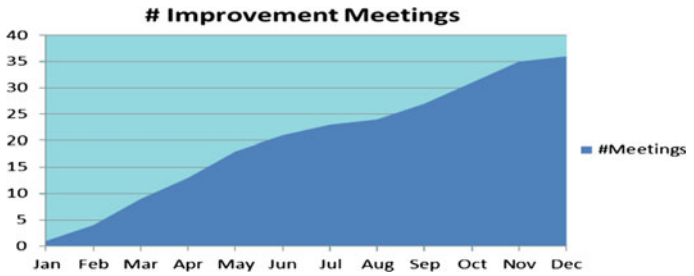


**Fig. 10.9** Process improvement productivity

Figure 10.9 gives an indication of the productivity of the improvement programme, and it shows how often the team meets to discuss the improvement suggestions and to act upon them. This chart is slightly naive as it just tracks the number of improvement meetings that have taken place during the year, and it has no information on the actual productivity of the meeting. The chart could be considered with Figs. 10.6, 10.7 and 10.8, to get more accurate information on the productivity of the team.

There will usually be other charts associated with an improvement programme, for example, a metric to indicate the status of the CMMI programme is provided in Fig. 10.26. Similarly, a measure of the current status of an ISO 9000 implementation could be derived from the number of actions which are required to implement ISO 9000, the number implemented and the number outstanding.

### 10.4.3 Human Resources and Training Metrics

These metrics give visibility into the human resources and training areas of a company. They provide visibility into the current headcount (Fig. 10.10) of the organization per calendar month and the turnover of staff in the organization (Fig. 10.11). The human resources department will typically maintain

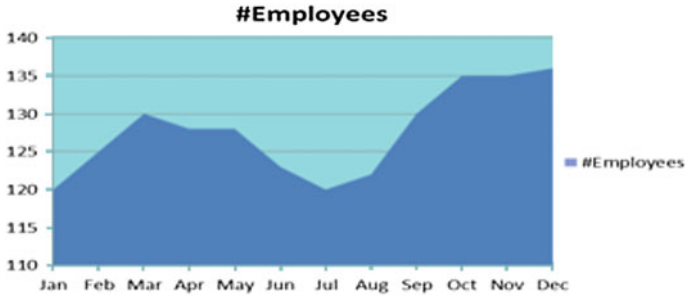


**Fig. 10.10** Employee headcount in current year


**Fig. 10.11** Employee turnover in current year

measurements of the number of job openings to be filled per month, the arrival rate of resumes per month, the average number of interviews to fill one position, the percentage of employees that have received their annual appraisal, etc.

The key goals of the HR department are defined and the questions and metrics are associated with the key goals. For example, one of the key goals of the HR department is to attract and retain the best employees, and this breaks down into the two obvious subgoals of attracting the best employees and retaining them. The next chart gives visibility into the turnover of staff during the calendar year. It indicates the effectiveness of staff retention in the organization.

### 10.4.4 Project Management Metrics

The goal of project management is to deliver a high-quality product that is fit for purpose on time and on budget. The project management metrics provide visibility into the effectiveness of the project manager in delivering the project on time, on budget and with the right quality.

The timeliness metric provides visibility into the extent to which the project has been delivered on time (Fig. 10.12), and the number of months over or under

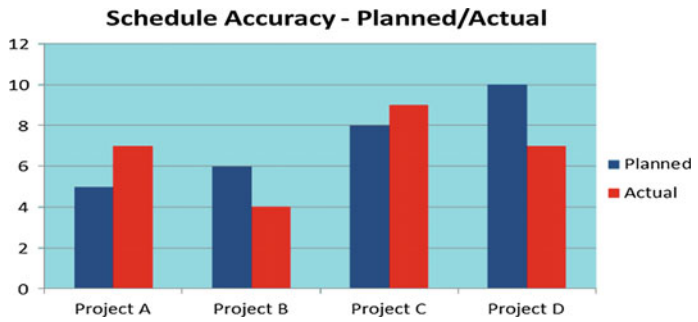


**Fig. 10.12** Schedule timeliness metric

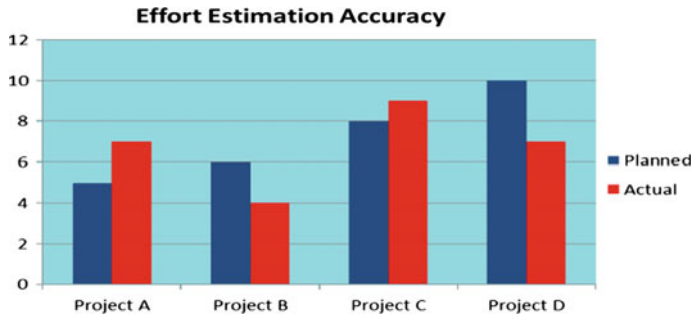


**Fig. 10.13** Effort timeliness metric

schedule per project in the organization is shown. The schedule timeliness metric is a lagging measure, as it indicates that the project has been delivered within schedule or not after the event.

The on-time delivery of a project requires that the various milestones in the project be carefully tracked and corrective actions are taken to address slippage in milestones during the project.

The second metric provides visibility into the effort estimation accuracy of a project (Fig. 10.13). Effort estimation is a key component in calculating the cost of a project and in preparing the schedule, and its accuracy is essential. We mentioned the Standish research data on projects in an earlier chapter, and this report showed that accurate effort and schedule estimation is difficult.

The effort estimation chart is similar to the schedule estimation chart, except that the schedule metric is referring to time as recorded in elapsed calendar months, whereas the effort estimation chart refers to the planned number of person months required to carry out the work, and the actual number of person months that it actually took. Projects need an effective estimation methodology to enable them to be successful in project management, and the project manager will use metrics to determine how accurate the estimation has actually been.

The next metric is related to the commitments that are made to the customer with respect to the content of a particular release, and it indicates the effectiveness of the projects in delivering the agreed requirements to the customer (Fig. 10.14). This chart could be adapted to include enhancements or fixes promised to a customer for a particular release of a software product.

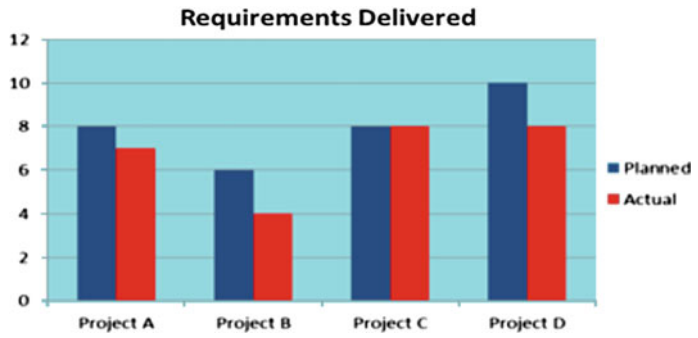


**Fig. 10.14** Requirements delivered

### 10.4.5 Development Quality Metrics

These metrics give visibility into the development and testing of the software product, and we presented a sample of testing metrics in Chap. 7. Figure 10.15 gives an indication of the quality of the software produced and the quality of the definition of the initial requirements. It shows the total number of defects and the total number of change requests raised during the project, as well as details on their severities. The presence of a large number of change requests suggests that the initial definition of the requirement was incomplete and that there is considerable room for improvement in the requirements elicitation process.

Figure 10.16 gives the status of open issues with the project, which gives an indication of the current quality of the project, and the effort required to achieve the desired quality in the software. This chart is not used in isolation, as the project manager will need to know the arrival rate of problems to determine the stability of the software product.

The organization may decide to release a software product with open problems, provided that the associated risks with the known problems can be managed. It is

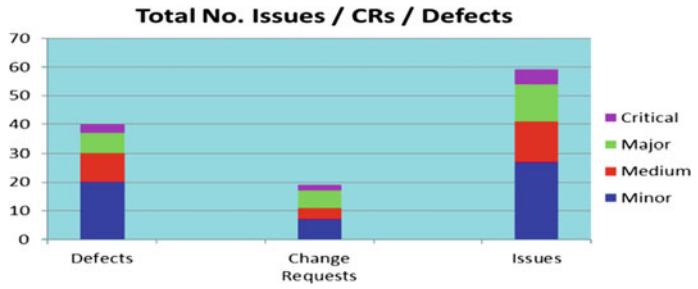


**Fig. 10.15** Total number of issues in project

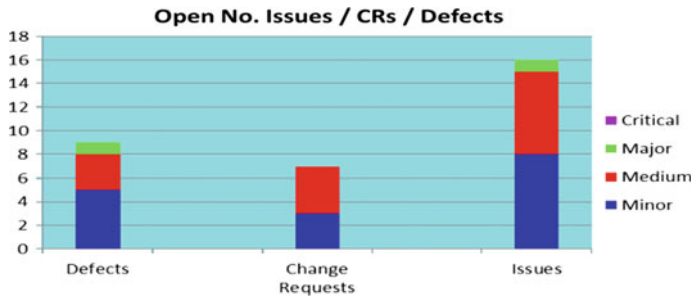


**Fig. 10.16** Open issues in project

important to perform a risk assessment to ensure that these may be managed, and the known problems (and workarounds) should be documented in the release notes for the product.

The project manager will need to know the age of the open problems to determine the effectiveness of the project team in resolving problems in a timely manner. Figure 10.17 presents the age of the open defects, and it highlights the fact that there is one major problem that has been open for over one year. The project manager needs to prevent this situation from arising, as critical and major problems need to be swiftly resolved.

The problem arrival rate enables the project manager to judge the stability of the software, and this (with other metrics) helps in judging whether the software is fit for purpose and ready for release to potential customers. Figure 10.18 presents a sample problem arrival chart, and the chart indicates positive trends with the arrival rate of problems falling to very low levels.

The project manager will need to do analysis to determine whether there are other causes that could contribute to the fall in the arrival rate; for example, it may be the case that testing was completed in September, which would mean, in effect, that no testing has been performed since then, with an inevitable fall in the number of problems reported. The important point is not to jump to a conclusion based on a particular chart, as the circumstances behind the chart must be fully known and taken into account in order to draw valid conclusions.

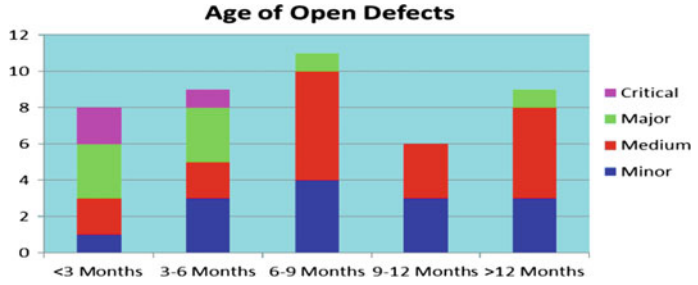


**Fig. 10.17** Age of open defects in project

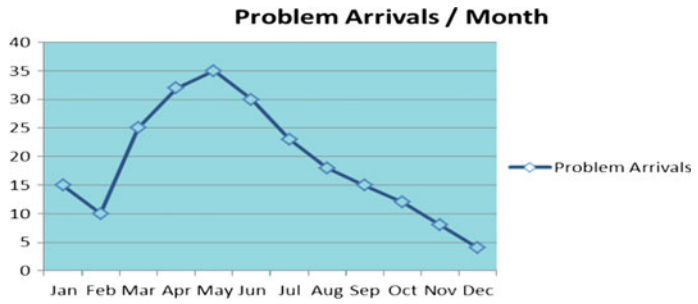


**Fig. 10.18** Problem arrivals per month

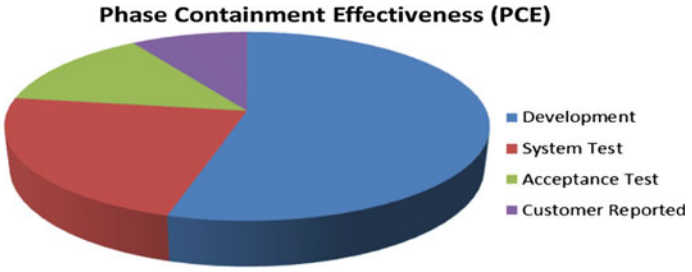


**Fig. 10.19** Phase containment effectiveness

Figure 10.19 measures the effectiveness of the project in identifying defects in the development phase and the effectiveness of the test groups in detecting defects that are present in the software. The development portion typically includes defects reported on inspection forms and in unit testing.

The various types of testing (e.g. unit, system, performance, usability, acceptance) were discussed in Chap. 7. Figure 10.19 indicates that the project had a phase containment effectiveness of approximately 54%. That is, the developers identified 54% of the defects, the system-testing phase identified approximately 23% of the defects, acceptance testing identified approximately 14% of the defects and the customer identified approximately 9% of the defects. The objective is that the number of defects reported at acceptance test and after the product is officially released to customer should be minimal.

### 10.4.6 Quality Audit Metrics

These metrics provide visibility into the audit programme and include metrics for the number of audits planned and performed (Fig. 10.20), and the status of the audit actions (Fig. 10.21). Figure 10.20 presents visibility into the number of audits carried out in the organization and the number of audits that remain to be done.


**Fig. 10.20** Annual audit schedule

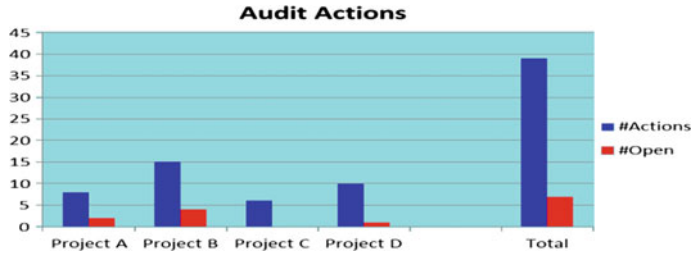


**Fig. 10.21** Status of audit actions

It shows that the organization has an audit programme and gives information on the number of audits performed during a particular time period. The chart does not give a breakdown into the type of audits performed, e.g. supplier audits, project audits and audits of particular departments in the organization, but it could be adapted to provide this information.

Figure 10.21 chart gives an indication of the status of the various audits performed. An auditor performs an audit and the results are documented in an audit report, and the associated audit actions need to be completed by the affected individuals and groups. Figure 10.21 presents the status of the audit actions assigned to the affected groups.

Figure 10.22 gives visibility into the type of actions raised during the audit of a particular area. They could potentially include entry and exit criteria, planning issues, configuration management issues, issues with compliance to the lifecycle or templates, traceability to the requirements, issues with the review of various deliverables, issues with testing or process improvement suggestions.

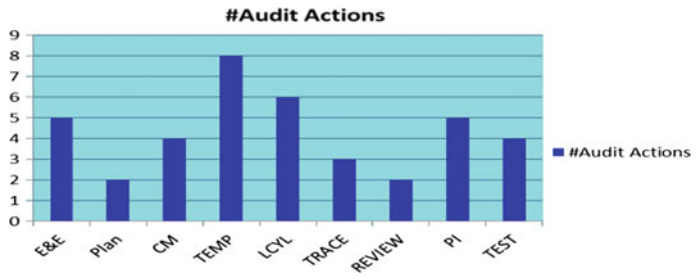


**Fig. 10.22** Audit action types

### 10.4.7 Customer Care Metrics

The goals of the customer care group in an organization are to respond efficiently and effectively to customer problems, to ensure that their customers receive the highest standards of service from the company and to ensure that its products function reliably at the customer's site. The organization will need to know its efficiency in resolving customer queries, the number of customer queries, the availability of its software systems at the customer site and the age of open queries. A customer query may result in a defect report in the case of a problem with the software.

Figure 10.23 presents the arrival and closure rate of customer queries (it could be developed further to include a severity attribute for the query). Quantitative goals are generally set for the resolution of queries (especially in the case of service level agreements). A chart for the age of open queries (similar to Fig. 10.17) is generally maintained. The organization will need to know the status of the backlog of open queries per month, and a simple trend graph would provide this. Figure 10.23 shows that the arrival rate of queries: in the early part of the year exceeds the closure rate of queries per month. This indicates an increasing backlog that needs to be addressed.

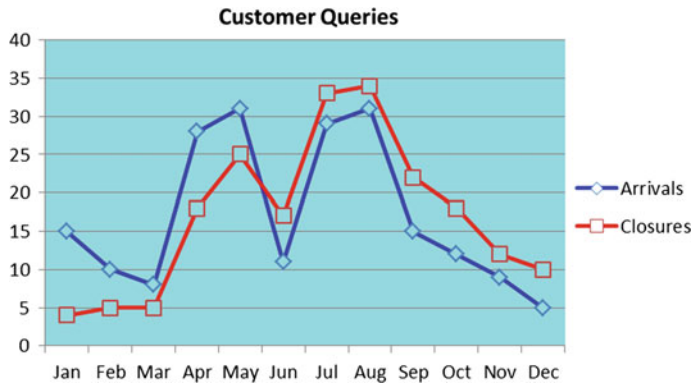


**Fig. 10.23** Customer queries (arrivals/closures)

The customer care department responds to any outages and ensures that the outage time is kept to a minimum. Many companies set ambitious goals for network availability: for example, the "*five nines initiative*" has the objective of developing systems which are available 99.999% of the time, i.e. approximately five minutes of downtime per year. The calculation of availability is from the formula:

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

where the mean time between failures (MTBF) is the average length of time between outages.

$$\text{MTBF} = \frac{\text{Sample interval time}}{\#\text{Outages}}$$

The formula for MTBF above is for a single system only, and the formula is adjusted when there are multiple systems.

$$\text{MTBF} = \frac{\text{Sample interval time}}{\#\text{Outages}} * \#\text{Systems}$$

The mean time to repair (MTTR) is the average length of time that it takes to correct the outage, i.e. the average duration of the outages that have occurred, and it is calculated from the following formula:

$$\text{MTTR} = \frac{\text{Total outage time}}{\#\text{Outages}}$$

Figure 10.24 presents outage information on the customers impacted by the outage during the particular month, and the extent of the impact on the customer.

An effective customer care department will ensure that a post-mortem of an outage is performed to ensure that lessons are learned to prevent a reoccurrence. This causal analysis details the root causes of the outage, and corrective actions are

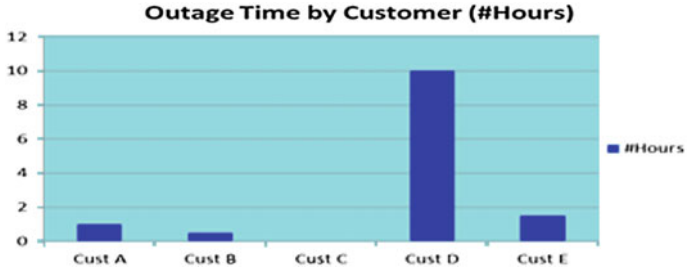


**Fig. 10.24** Outage time per customer


**Fig. 10.25** Availability of system per month

implemented to prevent a reoccurrence. Metrics to record the amount of system availability and outage time per month will typically be maintained by the customer care group in the form of a trend graph.

Figure 10.25 provides visibility on the availability of the system at the customer sites, and many organizations are designing systems to be available 99.999% of the time. System availability and software reliability are discussed in more detail in Chap. 11.

### 10.4.8 Miscellaneous Metrics

Metrics may be applied to many other areas in the organization. This section includes metrics on the CMMI maturity of an organization (where an organization is implementing the CMMI), configuration management and the cost of poor quality. Figure 10.26 gives visibility into the time to create a software release from the configuration management system.

The internal CMMI maturity of the organization is given by Fig. 10.27, and this chart is an indication of its readiness for a formal CMMI assessment. A numeric score of 1–10 is used to rate each process area, and a score of 7 or above indicates that the process area is satisfied.

Crosby argued that the most meaningful measurement of quality is the cost of poor quality [4] and that the emphasis on the improvement activities in the

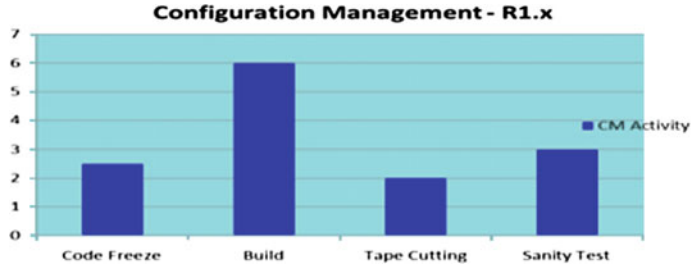


**Fig. 10.26** Configuration management

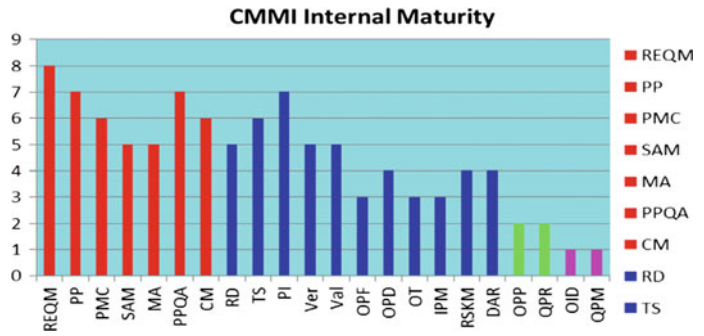


**Fig. 10.27** CMMI maturity in current year

**Table 10.2** Cost of quality categories

| Type of cost | Description |
|---|---|
| Cost external | This includes the cost of external failure and includes engineering repair, warranties and a customer support function |
| Cost internal | This includes the internal failure cost and includes the cost of reworking and retesting of any defects found internally |
| Cost prevention | This includes the cost of maintaining a quality system to prevent the occurrence of problems and includes the cost of software quality assurance and the cost of training. |
| Cost appraisal | This includes the cost of verifying the conformance of a product to the requirements and includes the cost of provision of software inspections and testing processes |

organization should therefore be to reduce the *cost of poor quality* (COPQ). The cost of quality includes the cost of external and internal failure, the cost of providing an infrastructure to prevent the occurrence of problems and the cost of the infrastructure to verify the correctness of the product.

The cost of quality was divided into four subcategories (Table 10.2) by Feigenbaum in the 1950s and evolved further by James Harrington of IBM.

The cost of quality graph (Fig. 10.28) will initially show high external and internal costs and low prevention costs, and the total quality costs will be high. However, as an effective quality system is put in place and becomes fully operational, there will be a noticeable decrease in the external and internal cost of quality and a gradual increase in the cost of prevention and appraisal.

The total cost of quality will substantially decrease, as the cost of provision of the quality system is substantially below the cost of internal and external failure. The COPQ curve will indicate where the organization is in relation to the cost of poor quality, and the organization will need to execute its improvement plan to put an effective quality management system in place to minimize the cost of poor quality.

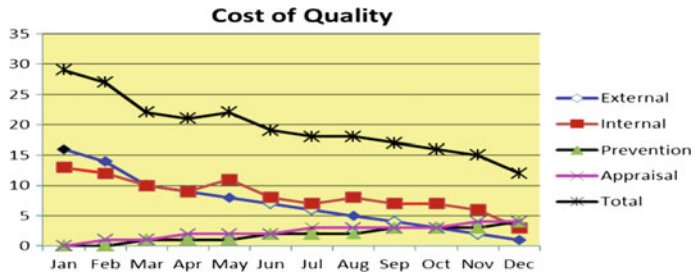


**Fig. 10.28** Cost of poor quality (COPQ)

## 10.5 Implementing a Metrics Programme

The metrics discussed in this chapter may be adapted and tailored to meet the needs of organizations. The metrics are only as good as the underlying data, and good data gathering is essential. The following are typical steps in the implementation of a metrics programme (Table 10.3):

The business goals are the starting point in the implementation of a metrics programme, as there is no sense in measurement for the sake of measurement, and so metrics must be closely related to the business goals. The next step is to identify the relevant questions to determine the extent to which the business goal is being satisfied, and to define metrics that provide an objective answer to the questions.

The organization defines its business goals, and each department develops specific goals to meet the organization's goals. Measurement will indicate the extent to which specific goals are being achieved, and good data gathering and recording are essential. First, the organization will need to determine which data needs to be gathered and to determine methods by which the data may be recorded. The information that is needed to answer the questions related to the goals will determine the precise data to be recorded. A small organization may decide to record the data manually, but often automated or semi-automated tools will be employed. It is essential that the data collection and extraction is efficient, as otherwise the metrics programme is likely to fail.

**Table 10.3** Implementing metrics

| Implementing metrics in organization |
|---|
| Define the business goals |
| Determine the pertinent questions |
| Define the metrics |
| Identify tools to (semi-) automate metrics |
| Determine data that needs to be gathered |
| Identify and provide needed resources |
| Gather data and prepare metrics |
| Communicate the metrics and review monthly |
| Provide training |

The roles and responsibilities of staff with respect to the implementation and day-to-day operation of the metrics programme need to be defined. Training is needed to enable staff to perform their roles effectively. Finally, a regular management review is needed, where the metrics and trends are presented, and actions identified and carried out to ensure that the business goals are achieved.

### 10.5.1 Data Gathering for Metrics

Metrics are only as good as the underlying data, and so data gathering is a key activity in a metrics programme. The data to be recorded will be closely related to the questions, and the data is used to give an objective answer to the questions. The business goals are usually expressed quantitatively for extra precision, and Table 10.4 presents an example of how the questions related to a particular goal are identified.

Table 10.5 is designed to determine the effectiveness of the software development process and to enable the above questions to be answered. It includes a column for inspection data that records the number of *defects* recorded at the various inspections. The *defects* include the phase where the defect originated; for example, a defect identified in the coding phase may have originated in the requirements or design phase. This data is typically maintained in a spreadsheet, e.g. Excel (or a dedicated tool), and it needs to be kept up to date. It enables the phase containment effectiveness (PCE) to be calculated for the various phases.

**Table 10.4** Goals and questions

| Goal | Reduce escaped defects from each lifecycle phases by 10% |
|---|---|
| Questions | How many defects are identified within each lifecycle phase?<br>How many defects are identified after each lifecycle phase is exited?<br>What percentage of defects escaped from each lifecycle phase? |

**Table 10.5** Phase containment effectiveness

| Phase of origin | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Phase | Inspect defects | Reqs | Design | Code | Accept test | In-phase defects | Other defects | % PCE |
| Reqs | 4 | | 1 | 1 | | 4 | 6 | 40 |
| Design | 3 | | | | | 3 | 4 | 42 |
| Code | 20 | | | | | 20 | 15 | 57 |
| Unit test | | 2 | 2 | 10 | | | | |
| System test | | 2 | 2 | 5 | | | | |
| Accept test | | | | | | | | |

We will distinguish between a defect that is detected *in-phase* and a defect that is detected *out-of-phase*. An in-phase defect is a problem that is detected in the phase in which it is created (e.g. usually by a software inspection). An out-of-phase defect is detected in a later phase (e.g. a problem with the requirements may be discovered in the design phase, which is a later phase from the phrase in which it was created).

The effectiveness of the requirements phase in Table 10.5 is judged by its success in identifying defects as early as possible, as the cost of correction of a requirements defect increases the later in the cycle that it is identified. The requirements PCE is calculated to be 40%, i.e. the total number of defects identified in phase divided by the total number of defects identified. There were four defects identified at the inspection of the requirements, and six defects were identified outside of the requirements phase: one in the design phase, one in the coding phase, two in the unit testing phase and two at the system-testing phase, i.e. 4/10 = 40%. Similarly, the code PCE is calculated to be 57%.

The overall PCE for the project is calculated to be the total number of defects detected in phase in the project divided by the total number of defects, i.e. 27/52 = 52%. Table 10.4 is a summary of the collected data and its construction consists of the following:

- Maintain inspection data of requirements, design and code inspections.
- Identify defects in each phase and determine their phase of origin.
- Record the number of defects in each phase per phase of origin.

The staff who perform inspections need to record the problems identified, whether it is a defect, and its phase of origin. Staff will need to be appropriately trained to do this consistently.

The above is just one example of data gathering, and in practice, the organization will need to collect various data to enable it to give an objective answer to the extent that the particular goal is being satisfied.

## 10.6 Problem-Solving Techniques

Problem-solving is a key part of quality improvement, and a *quality circle* (or problem-solving team) is a group of employees who do similar work and volunteer to come together on company time to identify and analyse work-related problems. Quality circles were first proposed by Ishikawa in Japan in the 1960s.

Various tools that assist problem-solving include *process mapping*, *trend charts*, *bar charts*, *scatter diagrams*, *fishbone diagrams*, *histograms*, *control charts* and *Pareto charts* [5]. These provide visibility into the problem and help to quantify the extent of the problem. The main features of a problem-solving team include:

- Group of employees who do similar work.
- Voluntarily meet regularly on company time.

- Supervisor as leader.
- Identify and analyse work-related problems.
- Recommend solutions to management.
- Implement solution where possible.

The facilitator of the quality circle coordinates the activities, ensures that the team leaders and team members receive sufficient training and obtains specialist help where required. The quality circle facilitator has the following responsibilities:

- Focal point of quality circle activities.
- Train circle leaders/members.
- Coordinate activities of all the circle groups.
- Assist in intercircle investigations.
- Obtain specialist help when required.

The circle leaders receive training in problem-solving techniques and are responsible for training the team members. The leader needs to keep the meeting focused and requires skills in team building. The steps in problem-solving include:

- Select the problem.
- State and restate the problem.
- Collect the facts.
- Brainstorm.
- Choose course of action.
- Present to management.
- Measurement of success.

The benefits of a successful problem-solving culture in the organization include:

- Savings of time and money.
- Increased productivity.
- Reduced defects.
- Fire prevention culture.

Various problem-solving tools are discussed in the following sections.

### 10.6.1 Fishbone Diagram

This well-known problem-solving tool consists of a cause-and-effect diagram that is in the shape of the backbone of a fish. The objective is to identify the various causes of some particular problem, and then, these causes are broken down into a number of subcauses. The various causes and subcauses are analysed to determine the root cause of the particular problem, and actions to address the root cause are then

defined to prevent a reoccurrence of the manifested effect. There are various categories of causes, and these may include people, methods and tools, and training.

The great advantage of the fishbone diagram is that it offers a crisp mechanism to summarize the collective knowledge that a team has about a particular problem, as it focuses on the causes of the problem, and facilitates the detailed exploration of the causes.

The construction of a fishbone diagram involves a clear statement of the particular effect, and the effect is placed at the right-hand side of the diagram. The major categories of cause are drawn on the backbone of the fishbone diagram; brainstorming is used to identify causes; and these are then placed in the appropriate category. For each cause identified, the various subcauses may be identified by asking the question "*Why does this happen?*" This leads to a more detailed understanding of the causes and subcauses of a particular problem.

*Example 10.1* An organization wishes to determine the causes of a high number of customer reported defects. There are various categories that may be employed such as people, training, methods, tools and environment. In practice, the fishbone diagram in Fig. 10.29 would be more detailed than that presented, as subcauses would also be identified by a detailed examination of the identified causes. The root cause(s) are determined from detailed analysis.

This example suggests that the organization has significant work to do in several areas and that an improvement programme is required. The improvements needed include the implementation of a software development process and a software test process; the provision of training to enable staff to do their jobs more effectively; and the implementation of better management practices to motivate staff and to provide a supportive environment for software development.

The causes identified may be symptoms rather than actual root causes: for example, high staff turnover may be the result of poor morale and a "blame culture", rather than a cause in itself of poor-quality software. The fishbone diagram

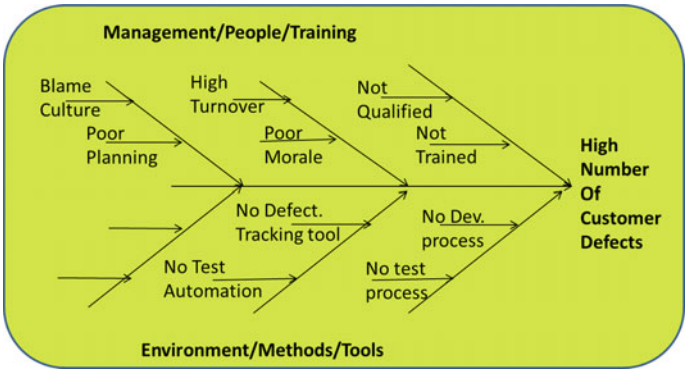


**Fig. 10.29** Fishbone cause-and-effect diagram

gives a better understanding of the possible causes of the high number of customer defects. A small subset of these causes is then identified as the root cause(s) of the problem following further discussion and analysis.

The root causes are then addressed by appropriate corrective actions (e.g. an appropriate software development process and test process are defined and providing training to all development staff on the new processes). The management attitude and organization culture will need to be corrected to enable a supportive software development environment to be put in place.

### 10.6.2 Histograms

A histogram is a way of representing data in bar chart format, and it shows the relative frequency of various data values or ranges of data values. It is typically employed when there are a large number of data values, and it gives a very crisp picture of the spread of the data values and the centring and variance from the mean.

The histogram has an associated shape; for example, it may be a *normal distribution*, a *bimodal* or *multi-modal distribution*, or be positively or negatively skewed. The variation and centring refer to the spread of data and the relation of the centre of the histogram to the customer requirements. The spread of the data is important as it indicates whether the process is too variable, or whether it is performing within the requirements. The histogram is termed process centred if its centre coincides with the customer requirements; otherwise, the process is too high or too low. A histogram enables predictions of future performance to be made, assuming that the future will resemble the past.

The construction of a histogram first requires that a frequency table be constructed, and this requires that the range of data values be determined. The data is divided into a number of data buckets, where a bucket is a particular range of data values, and the relative frequency of each bucket is displayed in bar format. The number of class intervals or buckets is determined, and the class intervals are defined. The class intervals are mutually disjoint and span the range of the data values. Each data value belongs to exactly one class interval and the frequency of each class interval is determined.

The histogram is a well-known statistical tool and its construction is made more concrete with the following example.

*Example 10.2* An organization wishes to characterize the behaviour of the process for the resolution of customer queries in order to achieve its customer satisfaction goal.

***Goal***
Resolve all customer queries within 24 h.
***Question***
How effective is the current customer query resolution process?
What action is required (if any) to achieve this goal?

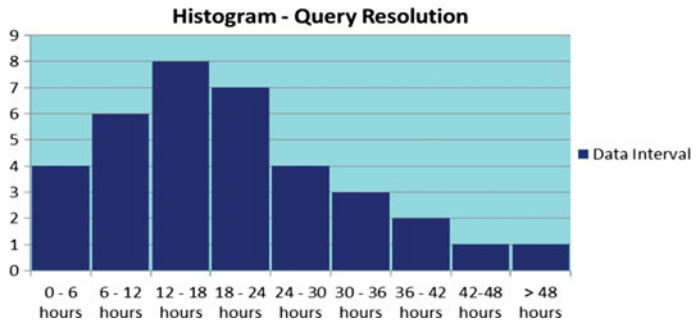


**Fig. 10.30** Histogram

The data class size chosen for the histogram below is six hours, and the data class sizes are of the same in standard histograms (they may be of unequal size for non-standard histograms). The sample mean is 19 h for this example. The histogram shown (Fig. 10.30) is based on query resolution data from 36 samples. The organization goal of customer resolution of all queries within 24 h is not met, and the goal is satisfied in (25/36 = 70% for this particular sample).

Further analysis is needed to determine the reasons why 30% of the goals are outside the target 24-h time period. It may prove to be impossible to meet the goal for all queries, and the organization may need to refine the goal to state that instead all critical and major queries will be resolved within 24 h.

### 10.6.3 Pareto Chart

The objective of a Pareto chart is to identify and focus on the resolution of problems that have the greatest impact (as *often 20% of the causes are responsible for 80% of the problems*). The problems are classified into various categories, and the frequency of each category of problem is determined. The Pareto chart is displayed in a descending sequence of frequency, with the most significant cause presented first, and the least significant cause presented last.

The Pareto chart is a key problem-solving tool, and a properly constructed chart will enable the organization to resolve the key causes of problems and to verify their resolution. The effectiveness of the improvements may be judged at a later stage from the analysis of new problems and the creation of a new Pareto chart. The results should show tangible improvements, with less problems arising in the category that was the major source of problems.

The construction of a Pareto chart requires the organization to decide on the problem to be investigated; to identify the causes of the problem via brainstorming; to analyse the historical or real time data; to compute the frequency of each cause; and finally to display the frequency in descending order for each cause category.

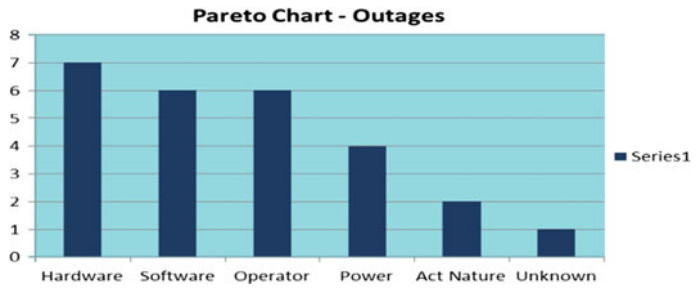


**Fig. 10.31** Pareto chart outages

*Example 10.3* An organization wishes to understand the various causes of outages and to minimize their occurrence.

The Pareto chart (Fig. 10.31) below includes data from an analysis of outages, where each outage is classified into a particular cause. The six causal categories identified are hardware, software, operator error, power failure, an act of nature and unknown. The three main causes of outages are hardware, software and operator error, and analysis is needed to identify appropriate actions to address these. The hardware category may indicate that there are problems with the reliability of the system hardware and that existing hardware systems may need improvement or replacement. There may be a need to address availability and reliability concerns with more robust hardware solutions.

The software category may be due to the release of poor-quality software, or to usability issues in the software, and this requires further investigation. Finally, operator issues may be due to lack of knowledge or inadequate training of the operators. An improvement plan needs to be prepared and implemented, and its effectiveness will be judged by a reduction in outages and reductions of problems in the targeted category.

### 10.6.4 Trend Graphs

A trend graph monitors the performance of a variable over time, and it allows trends in performance to be identified, as well as allowing predictions of future trends to be made (assuming that the future resembles the past). Its construction involves deciding on the variable to measure and to gather the data points to plot the data.

*Example 10.4* An organization plans to deploy an enhanced estimation process and wishes to determine whether estimation is actually improving with the new process.

The estimation accuracy determines the extent to which the actual effort differs from the estimated effort. A reading of 25% indicates that the project effort was

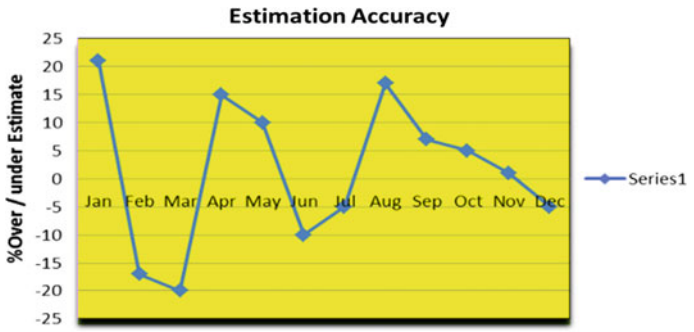


**Fig. 10.32** Trend chart estimation accuracy

25% more than estimated, whereas a reading of −10% indicates that the actual effort was 10% less than estimated.

The trend chart (Fig. 10.32) indicates that initially that estimation accuracy is very poor, but then, there is a gradual improvement coinciding with the implementation of the new estimation process.

It is important to analyse the performance trends in the chart. For example, the estimation accuracy for August (17% in the chart) needs to be investigated to determine the reasons why it occurred. It could potentially indicate that a project is using the old estimation process or that a new project manager received no training on the new process). A trend graph is useful for noting positive or negative trends in performance, with negative trends analysed and actions identified to correct performance.

### 10.6.5 Scatter Graphs

The scatter diagram is used to determine whether there is a relationship or correlation between two variables, and where there is to measure the relationship between them. The results may be a positive correlation, negative correlation, or no correlation. Correlation has a precise statistical definition, and it provides a precise mathematical understanding of the extent to which the two variables are related or unrelated.

The scatter graph provides a graphical way to determine the extent that two variables are related, and it is often used to determine whether there is a connection between an identified cause and the effect. The construction of a scatter diagram requires the collection of paired samples of data, and the drawing of one variable as the *x*-axis, and the other as the *y*-axis. The data is then plotted and interpreted.

*Example 10.5* An organization wishes to determine whether there is a relationship between the inspection rate and the error density of defects identified.

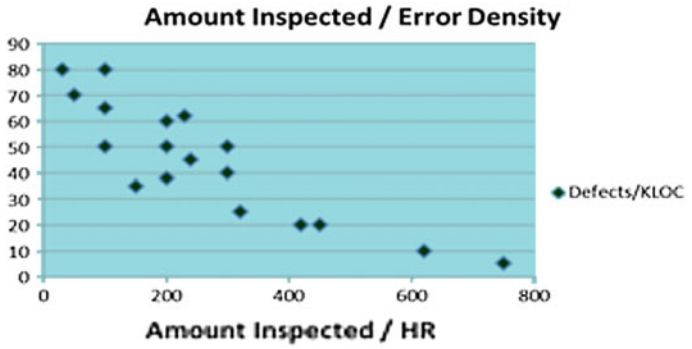


**Fig. 10.33** Scatter graph amount inspected rate/error density

The scatter graph (Fig. 10.33) provides evidence for the hypothesis that there is a relationship between the lines of code inspected and the error density recorded (per KLOC). The graph suggests that the error density of defects identified during inspections is low if the speed of inspection is too fast, and the error density is high if the speed of inspection is below 300 lines of code per hour. A line can be drawn through the data that indicates a linear relationship.

### 10.6.6 Metrics and Statistical Process Control

The principles of statistical process control (SPC) are important in the monitoring and control of a process. It involves developing a control chart, which is a tool that may be used to control the process, with upper and lower limits for process performance specified. The process is under control if it is performing within the lower and upper control limits.

Figure 10.34 presents an example on breakthrough in performance of an estimation process, and is adapted from [6]. The initial upper and lower control limits

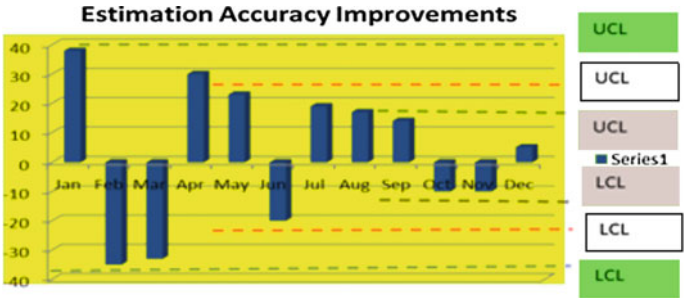


**Fig. 10.34** Estimation accuracy and control charts

for estimation accuracy are set at ±40%, and the performance of the process is within the defined upper and control limits.

However, the organization will wish to improve its estimation accuracy and this leads to the organization's revising the upper and lower control limits to ±25%. The organization will need to analyse the slippage data to determine the reasons for the wide variance in the estimation, and part of the solution will be the use of enhanced estimation methods in the organization. In this chart, the organization succeeds in performing within the revised control limit of ±25%, and the limit is revised again to ±15%.

This requires further analysis to determine the causes for slippage and further improvement actions are needed to ensure that the organization performs within the ±15% control limit.

## 10.7 Review Questions

1. Describe the Goal, Question, Metric model.
2. Explain how the Balanced Scorecard may be used in the implementation of organization strategy.
3. Describe various problem-solving techniques.
4. What is a fishbone diagram?
5. What is a histogram and describe its applications?
6. What is a scatter graph?
7. What is a Pareto chart? Describe its applications.
8. Discuss how a metrics programme may be implemented.
9. What is statistical process control?

## 10.8 Summary

Measurement is an essential part of mathematics and the physical sciences, and it has been successfully applied to the software engineering field. The purpose of a software measurement programme is to establish and use quantitative measurements to manage the software development processes in the organization, and to assist the organization in understanding its current software capability and to confirm that improvements have been successful.

This chapter includes a collection of sample metrics to give visibility into the various functional areas in the organization, including customer satisfaction metrics, process improvement metrics, project management metrics, HR metrics, development and quality metrics, and customer care metrics.

The Balanced Scorecard assists the organization in selecting appropriate measurements to indicate the success or failure of the organization's strategy. Each of the four scorecard perspectives includes objectives that need to be achieved for the strategy to succeed, and measurements indicate the extent to which the objectives are being met.

The Goal, Question, Metric paradigm is a rigorous, goal-oriented approach to measurement in which goals, questions, and measurements are closely integrated. The business goals are first defined, and then, the questions that relate to the achievement of the goal are identified, and for each question, a metric that gives an objective answer to the particular question is defined.

Metrics play a key role in problem-solving, and various problem-solving techniques were discussed. These include histograms, Pareto charts, trend charts and scatter graphs. The measurement data is used to assist the analysis, to determine the root cause of a particular problem and to verify that the actions taken to correct the problem have been effective. Trends may be seen over time, and the analysis of the trends allows action plans to be prepared for continuous improvement.

Metrics may be employed to track the quality, timeliness, cost, schedule and effort of software projects. They provide an internal view of the quality of the software product, but care is needed before deducing the behaviour that a product will exhibit externally.

## References

1. N. Fenton, *Software metrics: A Rigorous Approach* (Thompson Computer Press, 1995)
2. Victor Basili and H. Rombach, The TAME project. Towards improvement-oriented software environments. IEEE Trans. Softw. Eng. **14**(6) (1988)
3. R.S. Kaplan, D.P. Norton, *The Balanced Scorecard. Translating Strategy into Action* (Harvard Business School Press, 1996)
4. P. Crosby, *Quality is Free. The Art of Making Quality Certain* (McGraw Hill, 1979)
5. B. Michael, R. Diane, *The Memory Jogger. A Pocket Guide of Tools for Continuous Improvement and Effective Planning* (Goal *I* QPC, Methuen, MA, 1994)
6. G. Keeni et al., The evolution of quality processes at Tate Consulting Services. IEEE Softw. **17**(4) (2000)