Chapter 13

Organizational approaches to testing

Most of us live our lives in organizations that change routinely. We live in structures that are never completely finished. Software testing in particular is an area where companies try many different approaches to organization – partly because of the perplexity about which structures are most effective, i.e., lead to most effective relations between people.

Why do organizations exist? What are the most fundamental, basic building blocks of structural design? What are the advantages and disadvantages of the specific approaches used in the real world to organize software testing? How can we decide which approach to use at this stage of our company?

Organizations exist because there is a need for people as a group to behave predictably and reliably. To achieve results, people need to cooperate with each other. Organizations are created to support, and more to the point, coerce the activities people engage in. Good organizations minimize the inevitable conflict between the needs of the organization and the needs of the individual.

For example, some software organizations have successfully created a positive culture that encourages and rewards migration of individuals from product development groups into independent test development groups equally as much as migration from test to development. In fact, the best cultures reward those who over time complete the full loop – start by spending a few years in development, then spend a few years in test, then return to development.

We have known a few exceptional people to repeat this entire loop twice over a span of about 10 years. The amazing thing is that these people are viewed as the most valued, the most prized resource by managers in both product development and test development. Talk about job security! Why? What better person to have in development than someone who knows how to design a test library? This developer will know how to create software that passes tests, because he or she knows how to think like a tester. And what better person to have designing tests than someone who knows how developers think?

There are many managers who unfortunately only see testing as a training ground for development, not a place to put their best developers. Too bad. I count myself among the managers who *will not hire* a person to do testing unless that person is qualified today to join the development group responsible for developing the same product that he or she would be testing. The good testing manager will encourage such a person to get a few years' experience in product development, and only then to consider moving to the testing group. Radical thinking? We don't think so!

Organizing and reorganizing testing

Once we have personally experienced several reorganizations, we learn to appreciate the need for organizational change to be very carefully planned. Let's face it – organizational change creates stress, disruption, demoralization, and confusion to varying degrees. The following 2200-year old quotation helps us to see how long the problems surrounding organizational change have been recognized and articulated:

> "We trained hard ... but it seemed that every time we were beginning to form up into teams we would be reorganized ... I was to learn later in life that we meet any new situation by reorganizing, and a wonderful method it can be for creating the illusion of progress while producing confusion, inefficiency, and demoralization."
>
> Petronius Arbiter, 210 BC

Isn't it frightening how relevant this thought from Petronius remains today?

An organization is a system that combines materials, knowledge, and methods in order to transform various kinds of inputs into valued outputs. Organizational structure is the responsibilities, authorities, and relationships, arranged in a pattern, through which the organization performs its functions. Structural choices include reporting hierarchies, job descriptions, and goals.

A test organization (usually called a test group) is a resource or set of resources dedicated to performing testing activities. As shops grow, the need for a dedicated, independent testing function becomes a more apparent necessity. It takes unbiased people to produce an unbiased measurement – testing must be done independently if it is to be fully effective in measuring software quality.

Organizational structures are like software. Neither are ever really completely finished. With time, the demands (external and internal) on the structure change – the capability of the existing structure to meet these demands decreases – making organizational redesign a repetitive activity. Likewise, with time, the requirements for a software system change. As with organizations, no matter how well it is designed at the beginning, eventually, if it is to

remain useful, software must evolve. *We must leave behind the innocent delusion that once we understand the problem the software system is supposed to solve, we can go off alone to build and test it in peace.*

Test management is difficult. The manager of the test group must have the:

- ability to understand and evaluate software test process, standards, policies, tools, training, and measures;
- ability to maintain a test organization that is strong, independent, formal, and unbiased;
- ability to recruit and retain outstanding test professionals;
- ability to lead, communicate, support, and control;
- time to provide the care needed to manage test groups.

Senior managers must also have the ability to recruit and retain outstanding test managers. In fact, this is usually where the big mistakes are made. The senior managers do not understand the list above enough to know how to evaluate a potential test manager against these requirements. The result is that the wrong person is often promoted into test management!

When making organizational changes that affect testing, senior management needs to understand the impact the change will have on test management's ability to meet the above requirements. Plenty can go wrong when reorganizations occur without sufficient thought being given to the impact on testing. The dangers of having the wrong software testing structure include the following:

- Test independence, formality, and bias is weakened or eliminated.
- People in testing do not participate in reward programs.
- Testing becomes understaffed.
- Testing becomes improperly staffed with too many junior individuals.
- Testing is managed by far too junior managers.
- There is no leverage of test knowledge, training, tools, and process.
- Testing lacks the ability to stop the shipment of poor quality products.
- There is a lack of focused continuous improvement.
- Management lacks the bandwidth to manage testing groups.
- The quality focus is not emphasized.
- Test managers become demoralized owing to lack of career growth.

The above list can be used as a checklist of items to consider when planning organizational modifications. They can become questions; for example: Does making this change weaken the independence of testing? Does it hurt test's ability to get quality resources? etc.

Structural design elements

There is a surprisingly short list of basic building elements from which to construct a structure. The structural design elements are:

(1) *Tall or flat* – There may be many levels between the chief executive officer and the person on the shipping floor (this would be a tall organization), or there might be very few levels (a flat organization). In the last decade, flat has become more popular; managers finding themselves in the middle of a tall organization are particularly vulnerable to losing their jobs.
(2) *Market or product* – The organization may be structured to serve different markets or different products.
(3) *Centralized or decentralized* – The organization may be centralized or decentralized. This is a key question for the test organization. We will examine this in depth later in this chapter.
(4) *Hierarchical or diffused* – The organization may be hierarchical, that is, organized according to successively higher levels of authority and rank. Or it may be diffused, which is widely spread or scattered or matrixed.
(5) *Line or staff* – The organization will have a certain mix of line and/or staff roles.
(6) *Functional or project* – The organization may have functional or project orientations.

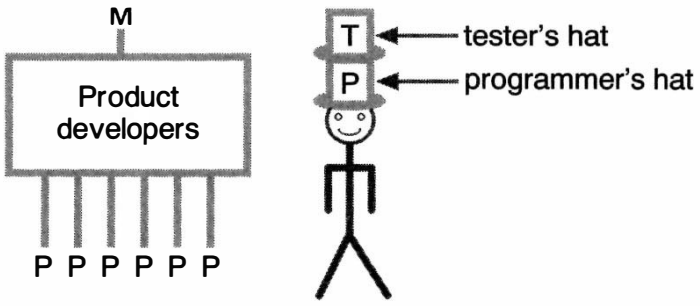
Combining these few design elements provides quite a variety of operating structural designs. Sometimes a variety of designs are implemented within the same company.

Approaches to organizing the test function

The above organizational basic design elements can be combined to produce many different test structures. There are seven approaches to organizing testing typically taken in practice that reflect the evolution of a maturing development organization. The following approaches to structure assume that unit testing should be done by product development. Therefore, the material that follows is about testing activities not related to unit testing, e.g., function testing, system testing, etc.

Approach 1. Testing is each person's responsibility

Approach 1 is what often occurs in the real world, especially when companies are small and little thought has been given to testing. As shown in Figure 13.1, there is a group of product developers whose primary responsibility is to



Advantages	Disadvantages
Seems on the surface like the most natural solution. (Evolves naturally out of the original product development team.)	Programmers are inherently incapable of effectively testing their own programs.

Figure 13.1 Approach 1: testing is each person's responsibility.

build a product. In this model, these product developers also are responsible for testing their own code. These people unfortunately must try their best to wear two hats, a programmer's hat and a tester's hat. They are responsible for function testing, system testing, and any other kind of testing that gets done.

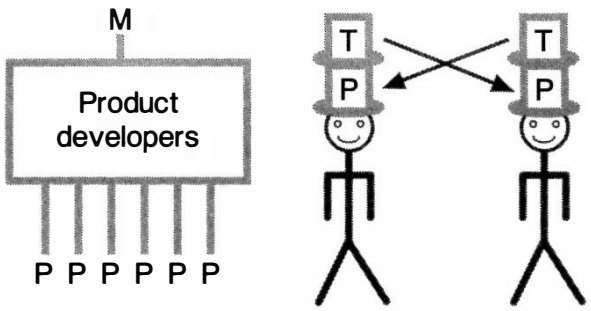
The problem with this approach is that it violates a basic assumption. Testing must be done independently if it is to be fully effective in measuring software quality. Programmers are biased by the creative work they do. This blinds them to their own errors; it is human nature.

Approach 2. Testing is each unit's responsibility

Approach 2 fixes the obvious flaw encountered in approach 1 – that of programmers being inherently incapable of testing their own programs – by assigning product developers within the group the job of testing each other's code. Note from Figure 13.2 that each person is still wearing two hats: the person on the right is responsible for product development of their own modules, plus they must test their team-mate's modules.

The problem now is for these people to find the time to understand the job of the software testing professional as well as understand product development processes, standards, policies, tools, training, metrics, etc. For typical software industry projects, this is just asking too much of one person. It is like expecting your average construction site hire to be a master electrician and a master carpenter on the same project at the same time. It is not impossible, it can occur – it is just not likely and does not make much sense.

In reality, these people will pick one hat to wear – the hat for which they know they have the primary responsibility and for which they are evaluated by management – the hat of the product developer. They will take time to test other people's code as time and skills permit. They will usually not get very good at learning the special skills, tools, and methods of testing while they are simultaneously responsible for developing product. That's reality.



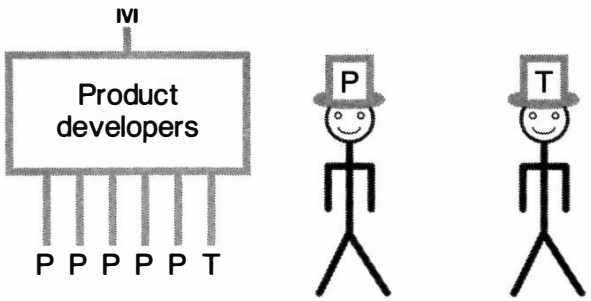
Advantages	Disadvantages
Solves the problem of programmers being incapable of testing their own code.	Programmer bandwidth (one hat too many). Methodologies, skills, training.

Figure 13.2 Approach 2: testing is each unit's responsibility.

Approach 3. Testing is performed by dedicated resource

Approach 3 solves the developer bandwidth issue by selecting a product developer and giving them a new job, that of test developer. Note from Figure 13.3 that each person in the group now only has to wear one professional hat. The tricky part here is for the group manager to pick the right person for testing.

In the evolutionary beginning, we noted that there was no test organization at all. One day the manager wakes up and declares, "All right, I see, I believe now. I know we need a dedicated person for testing. Hmmm, whom shall I tap to be our tester?" This really happens. Once while the author was presenting a course on testing, a development manager raised his hand and



Advantages	Disadvantages
Solves the developer bandwidth problem. (Only wearing one professional hat now). Single team.	Management bandwidth. Can management provide test process, standards, policies, tools, training, measures?

Figure 13.3 Approach 3: Testing performed by dedicated resource.

stated, "OK, OK, now I understand what testing is all about. I'm going to put our first full-time independent tester in place!" Great, thinks the course instructor, feeling very good about his teaching abilities.

"Yea", the manager continues, "I have this person that just doesn't fit on the development team. They've been a consistently poor performer; they're unhappy and they are just not contributing anything to reaching our project goals. I was going to fire them, but that is such an unpleasant task for me. I'll make them a tester! What have I got to lose?"

Of course we know he's got plenty to lose. The individual wearing too many hats now is the first line product development manager. This person, in addition to providing guidance to a product development team, must provide guidance on testing process, testing standards, testing policies, testing tools, testing training, testing measures, hiring professional testers, etc. It is just not going to happen.

It becomes clear that some sort of test group is necessary – a set of resources dedicated to performing testing activities and managed by a test manager. Without a formal organization, testing practices and tools must be set up for every project. With a separate test group, however, an organization remains in place to serve all projects on a continuing basis – to provide management with independent, unbiased, quality information.

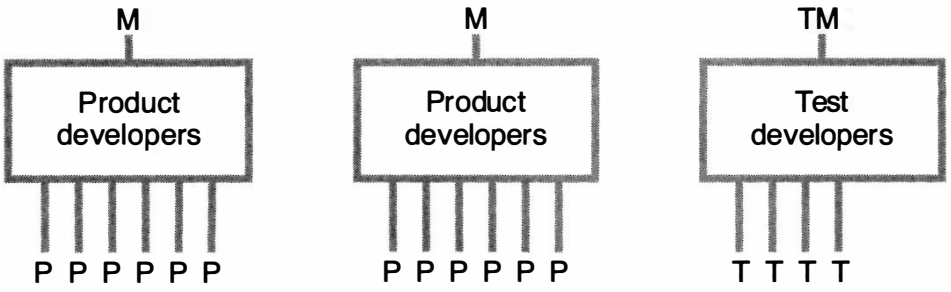
Bill Hetzel, in the book *The Complete Guide to Software Testing* (1988), tells us:

> An independent test organization is important because
>
> - building systems without one has not worked well,
> - effective measurement is essential to product quality control,
> - coordinating testing requires full-time, dedicated effort.

The creation of a formal test organization solves the problem of approach 3. Note in Figure 13.4 the importance of a test organization is realized, headed by a test manager. The question now becomes where to put the test group organizationally.

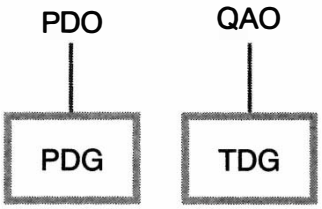
Approach 4. The test organization in QA

A common solution is to make the new test organization a component of quality assurance, where QA also audits the development process (see Figure 13.5). The manager of QA may not understand software testing. The capabilities of management of the testing group are critical, as is the manager of the testing manager. Since testing is performed in a separate organization from development, it will take extra effort to encourage and create a positive team environment. People also begin to ask, who owns quality? Development management complains that they no longer have the complete set of resources needed to produce a quality product.



Advantages	Disadvantages
Solves the management problems of approach 3.	Where to put it organizationally?

Figure 13.4 The test organization.



PDO = Product Development Organization
PDG = Product Development Group

QAO = Quality Assurance Organization
TDG = Test Development Group

Advantages	Disadvantages
Solves the management problems of approach 3 by creating a formal test organization which is a part of QA.	Possible teamwork problems. TDG can get lost in QA group with other responsibilities. The product development group is not wholly accountable for producing quality products.

Figure 13.5 Approach 4: the test organization in QA.

Approach 5. The test organization in development

Approach 5 attempts to deal with the teamwork and quality ownership issue identified in approach 4 by creating a test organization that is part of the development organization (see Figure 13.6). This approach usually puts the test group under the second line product development manager.

Unfortunately, this is asking too much of most second line product development managers. It is much less an issue for higher level managers such as vice presidents that understand the need for a strong, independent

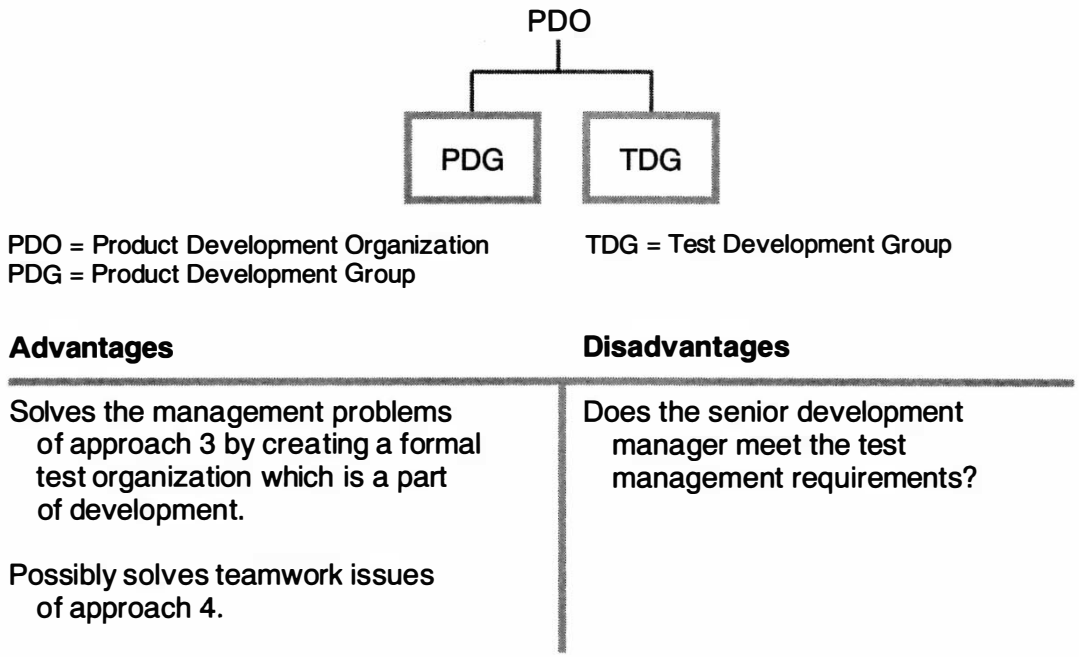


Figure 13.6 Approach 5: the test organization in development.

test function and who are willing to put a strong manager in place to manage the test group. In any event, all is riding on the senior manager who is now the one wearing two hats, that of managing the management responsible for product development and of managing the management responsible for software testing.

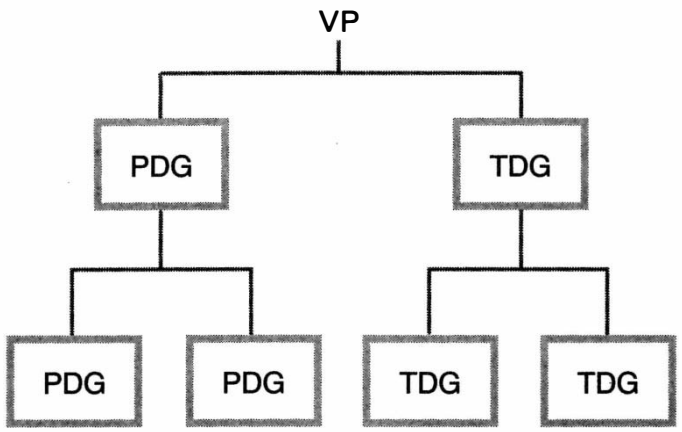
As organizations grow, more people are hired into the test organization, and multiple test groups are needed. A new issue arises. Should the multiple test groups be centrally or decentrally organized?

Approach 6. Centralized test organization

Approach 6 solves the senior management problem of approach 5 by creating a central test organization that lives within and serves a product development division (see Figure 13.7). Note how this creates a major opportunity for a senior test manager/director to significantly impact the organization. For example, the senior test manager can:

- manage the sharing of testing resources (both people and equipment) to smooth needs and better manage the risks of the company;
- coordinate consistent training for all testers across several test groups;
- promote consistent and high-quality testing tools for use by all testers;
- find and hire strong first line testing managers;
- provide real testing guidance to first line test managers.

In addition, first line testing managers see a potential career path in testing as they aspire to become a senior test manager.



VP = Vice president
PDG = Product Development Group
TDG = Test Development Group

Advantages	Disadvantages
Solves the senior management problems of approach 5 by centralizing the formal test organization which is a part of development.	Test organization lives or dies by VP.
Creates a career path for test managers.	Potential lack of teamwork at individual/project level.
	Possible lack of consistent test methodologies across VP organizations.

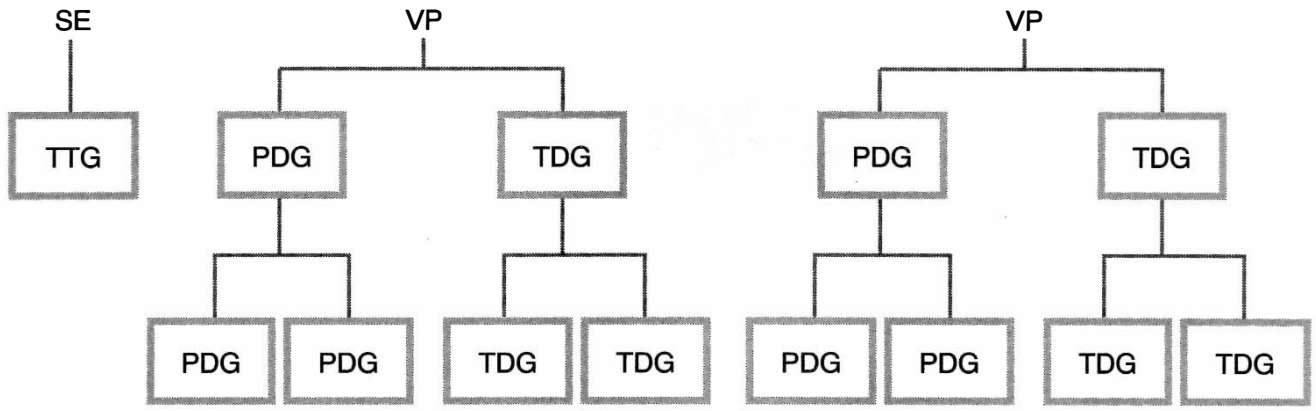
Figure 13.7 Approach 6: centralized test organization.

Now the risks are centralized, and assuming a good senior test manager is in place, the testing organization will live or die by the vice president who manages the product development and the test development organizations. When it's time to determine headcount, capital, and rewards for software testing, the vice president must provide the crucial support.

With the right vice president, this can and has often worked quite well in practice. A vice president that provides the proper resources, hires well, and delegates well to a highly competent senior test manager can make this approach work extremely well.

By creating project teams that are loaned resources from the independent test organization and that otherwise act, communicate, and live close together as a team alongside the product developers, the previously stated teamwork disadvantages are minimized. And when serious discussions regarding product quality are needed, a senior test manager is there to provide an unbiased voice to the vice president. This can be of great value to a vice president who cannot get the complete set of facts from product development management.

As companies continue to grow larger and contain multiple divisions, companies using this approach may begin to discover a lack of consistent approaches and best practices across different vice president organizations. This is solved by the next and final approach.



Advantages	Disadvantages
Solves the consistency problems of approach 6 by centralizing a test technology group which is a part of software engineering.	Test organizations live or die by VP. Potential lack of teamwork at individual/project level.

Figure 13.8 Approach 7: centralized and test technology.

Approach 7. Centralized test organization with a test technology center

Approach 7 solves the consistency problems of approach 6 by creating a test technology group, which in this case is part of a software engineering function (see Figure 13.8). This new test technology group is responsible for:

- leading and managing testing process and testing productivity improvement efforts;
- driving and coordinating testing training programs;
- coordinating the planning and implementation of testing tool programs;
- documenting test process, standards, policies, and guidelines as necessary;
- recognizing and leveraging best practices within the testing groups;
- recommending, obtaining consensus for, and implementing key testing measurements.

Selecting the right approach

How can we decide from among the many different approaches to using these structural design elements? To begin with, below is a set of sample selection criteria that can be used as a basis for evaluating different approaches. To what extent does the organizational structure:

- provide the ability for rapid decision making;
- enhance teamwork, especially between product development and testing development;
- provide for an independent, formal, unbiased, strong, properly staffed and rewarded, test organization;
- help to coordinate the balance of testing and quality responsibilities;
- assist with test management requirements as stated earlier in this chapter;
- provide ownership for test technology;
- leverage the capabilities of available resources, particularly people;
- positively impact morale and career path of employees (including managers)?

Providing for rapid decision making leads to improved responsiveness. It is possible to overcome an organizational structure that does not inherently provide for rapid decision making by creating special core teams and/or defining a special rapid escalation process, i.e., a daily high-level management short decision meeting used to escalate and deal with project issues.

One approach to doing reorganizations is to follow these steps:

(1) map the current organization;
(2) define and prioritize the key selection criteria (see list above);
(3) document new potential alternative approaches;
(4) evaluate potential approaches using a selection grid (see below);
(5) decide on a new organization;
(6) implement the new organization.

A decision matrix selection grid is a quality tool that can be used to evaluate the various approaches while using the selection criteria. Once the selection criteria are decided, they can be numbered or uniquely identified and placed across the top of the decision matrix.

The possible approaches are also numbered and placed vertically on the matrix as shown in Figure 13.9. Each approach is evaluated by considering each selection criteria and assigning a number to enter into the matrix. This number would come from a pre-defined range of values, e.g., from 1 to 5, where 5 indicates a high ranking, meaning the organization to a very great

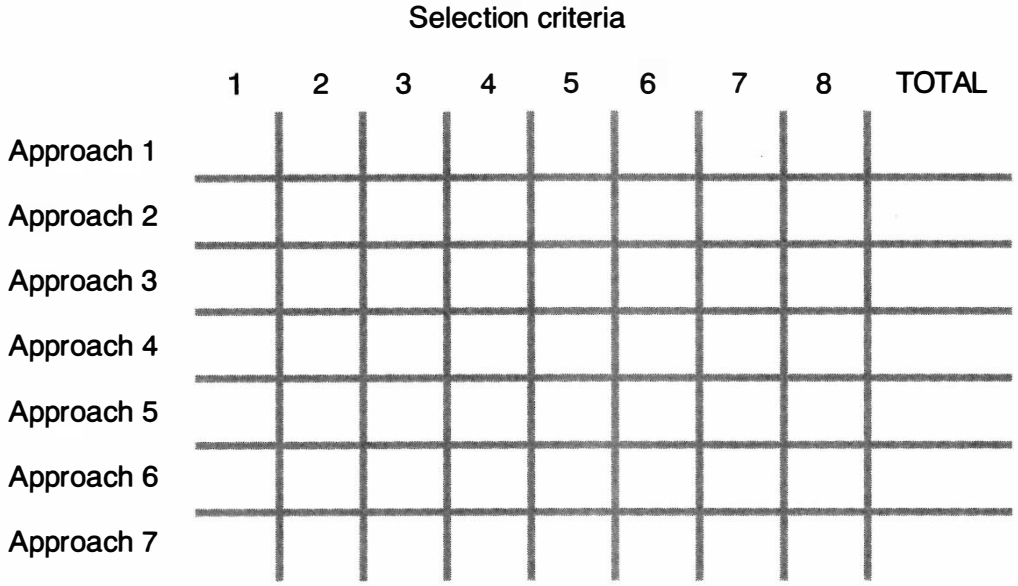


Figure 13.9 Decision matrix selection grid.

extent meets the given criteria for the approach under consideration. Likewise 3 might indicate a medium ranking and 1 a low ranking.

Add the scores horizontally to yield a total score for each approach. As a more advanced enhancement to this approach, the selection criteria can be assigned a multiplier weighting to be factored in.

Reorganizations should be carefully planned and implemented. One last piece of advice – remember to involve the participants in the above process!

References

Beizer, B. (1992). "Losing It, An Essay on World Class Competition," *Software Quality World*, **4**(3).

Goodman, Paul, Sproull & Associates (1990). *Technology and Organizations*. San Francisco, CA: Jossey-Bass.

Gelperin, D. and Hetzel, W. (1989). *STEP: Introduction and Summary Guide.* Jacksonville, FL: Software Quality Engineering.

Hetzel, W. (1988). *The Complete Guide to Software Process.* Wellesley, MA: QED Information Sciences.

Humphrey, W.S. (1989). *Managing the Software Process*. Reading, MA: Addison-Wesley.

Kaner, C. (1988). *Testing Computer Software.* Blue Ridge Summit, PA: TAB Books.

Kit, E. (1992). "Approaches to Organizing the Independent Test Function," *Software Testing Analysis & Review (STAR) Conference Proceedings.*

Silverman, M. (1984). *The Technical Manager's Survival Book.* New York: McGraw-Hill.