Organizing for Successful Software Development

BY: Marc Hamilton in conjunction with Harris Kern's Enterprise Computing Institute

Many CIO's recognize that the organizational structure of their software development group has an impact on the success of their application development efforts. Unfortunately, there is not always the same level of consensus between CIO's on what the correct organizational structure should be. By defining the organizational structure and examining its importance to successful software development through different types of organizational structures, along with their pros and cons. Sample organization charts are given small, medium, and large software development organizations. We'll discuss also centralized versus decentralized organizations and the use of virtual project teams.

The Dimensions of an Organization

An organization is defined by much more than boxes containing job titles and names connected by lines representing a reporting structure. Besides organizational structure, the multiple dimensions spanning the people, processes define an organization, and technology represented within it. Some of these dimensions include:

- ***People Dimensions***: Each individual in an organization has certain skills, and these skills are typically measured against formal or informal performance metrics leading to rewards (compensation) as incentives for future performance. The people in an organization establish its culture, those behavior patterns and values that are generally recognized as being adopted.

- ***Process Dimensions***: The procedures and methodologies used by people in the organization. Almost all organizations define their own internal economies through processes for budgeting, priority settings, and project approval.

- ***Technology Dimensions***: The specific skills and tools people in the organization use to carry out the business function of the organization.

The Importance of Organizational Structure

Of the different people, process, and technology dimensions of an organization, structure is by far the most fundamental. Without a sound structure, people in the organization lose their culture and compete for individual rewards rather than for the good of the organization. Without structure, processes have no home and internal economies collapse because of conflicting objectives. Without structure, technology is no longer pursued as a research interest rather than for the good of the organization. While these concepts hold true for any information technology organization, they are especially true for software development organizations, no matter what their size.

Many a small software startup begins life with no more than a couple of developers working out of a garage. Not much organizational structure is required at this point in a company's history, however organizational structure still exists. For instance, in 1977, when Bill Gates and Paul Allen formed their partnership and officially named it Microsoft, the company had minimal organizational structure. Less than a dozen employees worked at Microsoft's first office in Albuquerque, New Mexico, and everyone knew who was in charge. No complicated organization charts were needed to figure out everyone's reporting structure. At the same time, all employees knew what their role was in the company and what they were trying to accomplish. This was because any organizational structure that was needed could be informally communicated between each of the employees.

At the other end of the spectrum are IT departments of Fortune 500 companies, large independent software vendors, and commercial system integrators. An entry-level programmer at Microsoft today probably needs several different organizational charts to show the reporting structure among 20,000+ employees and up to Bill Gates. Having organization charts alone, unfortunately, is no guarantee of a healthy corporate structure. In any large organization, there are many dimensions to measuring the success of the corporation's structure. While no organizational model fits all development departments, certain traits stand out among companies that routinely produce successful software products.

Streamlining Bureaucracy

One of the side effects many development organizations have suffered as they grew has been increased bureaucracy. Although we focus on people and process issues, the aim is to help streamline bureaucracies, not develop new ones For instance, at some companies, a standard operating procedure is that, with few exceptions, no document shall ever require more than two approvals, one from the person who authored the document and one from the final approver. Besides empowering employees, this makes it very simple to place blame when an incorrect decision is made. In addition, this removes the possibility for two superiors to reject a document and send it back to the original author with conflicting modifications.

Sample Organizational Structures

The next four sections describe alternative organization schemes commonly found in software development departments:

- Project centered organizations
- Department centered organizations
- Matrix organizations
- Product line organizations

While none of the above will be perfect for every software development group, they each offer useful ideas for coming up with your organization scheme.

Project Centered Organizations

Organizations centered on project teams are typically found in smaller or newly formed groups. A project centered organization approach is suitable for groups of about 5 to 40 people supporting 1 to 8 projects of small to medium duration, perhaps up to a year each. In such an organization, each group is primarily self-sufficient and is staffed by enough skilled developers to address every stage of the development life cycle. This in turn means most individuals will have responsibility for some facet of development other than just programming, such as requirements, architecture, configuration management, or testing.

As development organizations grow larger, project centered organizations become less desirable. At one level, the number of projects grows to outstrip the needed specialty skills so you cannot provide a developer with the needed skills to each project. Another problem is specialist knowledge and even general skills tend to not be shared between individual projects that are operating in their own microcosms. As organizations outgrow project-centered organizations they often reorganize into department-centered organizations.

Department Centered Organizations

Department centered development organizations start to become practical as a group grows above 25 developers or 5 projects. At these staffing levels, there are sufficient people to form multiple departments centered on particular software skills or life cycle areas. For instance, a 40-person group might have departments for:

- System and database administrators
- User interface programmers
- Application programmers
- Configuration management, test, and quality assurance

A common mistake in department-centered organizations is to break software architects into a separate department or group. We have found this can lead to elitism and be very counterproductive. First, it starts to separate the architects from the developers who are doing the actual implementation. Architects thus become more quickly out-of-touch with the latest development methodologies actually being used. Also, while every developer does not want to be an architect, every developer likes to have some say in the design. If developers are too separated from architects, they may have a built-in incentive to prove the architect's design was wrong by not working there hardest to implement it. When this happens the architect will most likely blame the problem on developer incompetence than on any architectural flaws. The whole iterative development process becomes harder to implement smoothly.

Matrix Organization

When your development organization grows to several hundred people or more, you may want to consider a matrix organization. Matrix organizations are sometimes used in companies with a large number of software developers working on a broad array of software projects. One side of the matrix is organized along skill sets while the other side of the matrix is organized across projects. In a matrix organization, every developer has two managers. One manager is from the department or skill set matrix and one manager is from the project matrix. A developer typically stays in a single department for as long as he or she continues working in that skill area. A developer would only stay on a project for the length of time his or her particular skill was needed and then return to his or her department for another assignment. Table 1-1 shows how employees might be assigned from different departments to several different projects. As shown in the table, not all departments necessarily have employees working on all projects.

Table 1-1 Sample Personnel Assignments

Department	Project 1	Project 2	Project 3
Requirements Analysis	John	Kevin	Betty
Business Systems	Steve, Nancy	Carol	Bruce
Web Development	David		Barbara
Operating Systems	Charlie	Jeff	
Real-time Processing			Lisa
Configuration Management	Brian	Peter	Frank
Integration and Test	Joe, Henry	Dan, Tim	Leslie

Individual departments are responsible for hiring and training developers and supplying them to projects as needed. Department managers work with project managers to properly forecast requirements and equitably assign developers to projects considering the best interests of the corporation.

A matrix organization obviously requires a certain minimum size to sustain the overhead of two management chains. One challenge with such an organization is to develop the right number and mix of departments. Another challenge is to sustain developer loyalty to projects when their long-term management lies in the department organization. Because of these issues, a product line organization is often bettered suited.

Product Line Organizations

In a product line organization, developers are organized into projects based on business product lines as opposed to skill set departments. A product line organization is responsible for staffing the skill sets required for its project mix. For instance, one product line might have requirements analysts, OS experts, some web developers, and configuration management. Another product line might have requirements analysts, real-time coding

experts, and configuration management. This works if product line organizations are sufficiently large that enough developers exist to staff duplicated functions throughout departments. The downside is that software projects will most often require different sets of skill levels at different times in the software life cycle. Each product line must always have sufficient resources to staff for peak periods while worrying about the lulls in-between.

Recurring Organizational Themes

When choosing how to organize your software development organization, these recurring themes and concepts are ones you should address:

- Creating a software process team
- Balancing centralized versus decentralized organizations
- Managing virtual teams

Creating a Software Process Team

Regardless of organization, every development organization should have a software process team. This team, made up of representatives from each software skill area, should be tasked with developing standard processes used throughout the organization. These individuals can thus become process "experts" that help train the rest of the organization.

Balancing Centralized Vs. Decentralized Organization

Most IT groups have experimented with different mixes of centralized versus decentralized organizations. The arguments on both sides are well-known. Centralized organizations generate economies of scale and provide developers the most opportunity to specialize. A development group of 50 people can probably have several specialists in user interface development. Break that same organization down into 10 groups of five and no group may be able to afford a dedicated user interface or other specialist. The down side of centralized organizations are they often are not responsive to individual business unit demands, especially to smaller business units. In theory, a decentralized development group dedicated to an individual business unit can be more responsive to local needs.

Our suggestion is to keep a small centralized development organization for enterprise-wide systems and for company-wide architecture issues. Individual business units should be responsible for developing their own local applications.

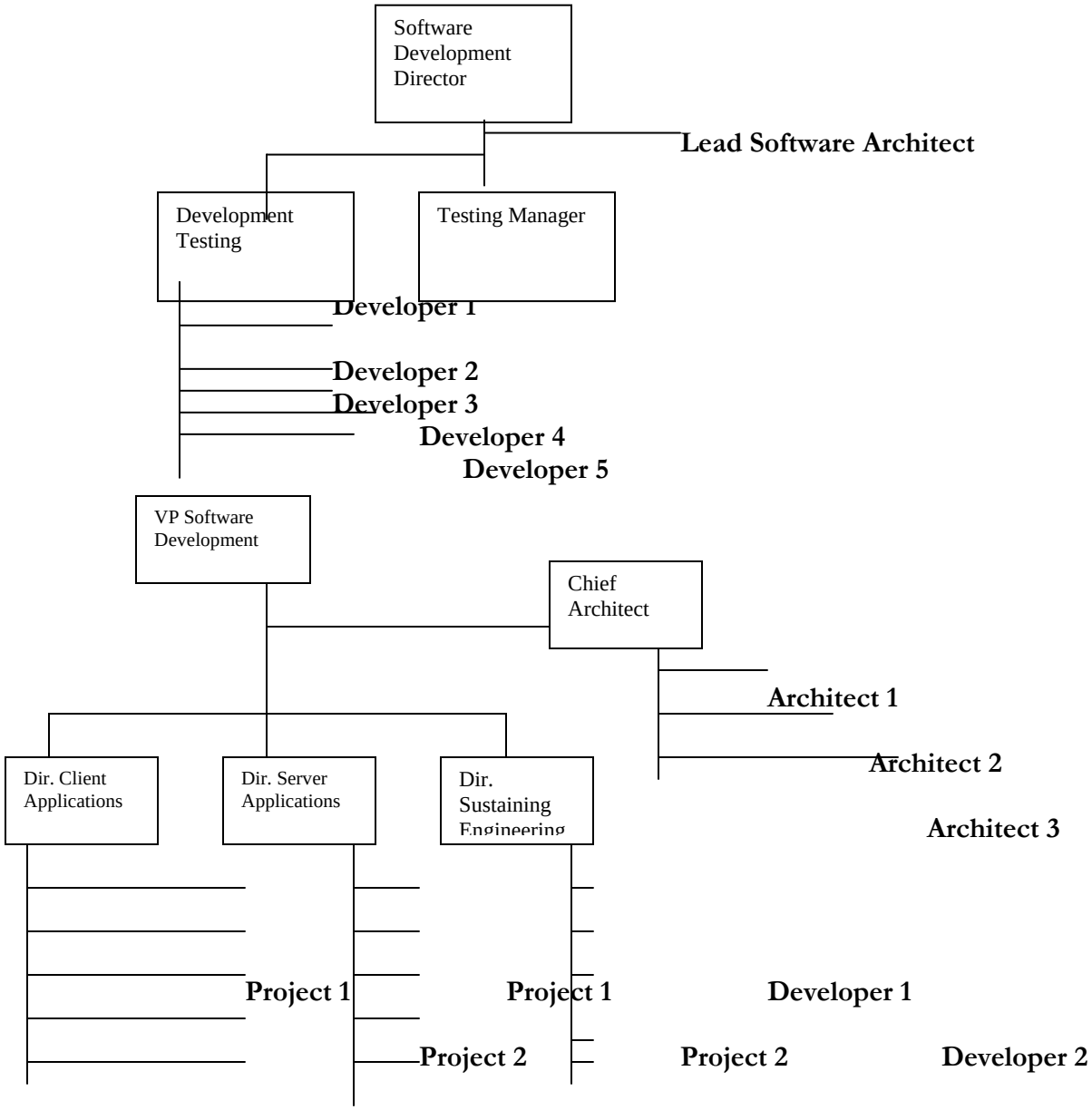
Managing Virtual Teams

Static software development organizations worked well when software was limited to a small, well-defined, and static set of functions within an organization. Today, business requirements often may call for the creation of virtual teams that span across all aspects of a

company, not just its development organization. The classic example is the marketing department that decides the company needs to have an electronic commerce web site. Besides the IT and marketing departments, this might involve the legal department, the sales department, product departments, and the art department. Today's business drivers mean such teams need to be able to come together, perform their function, turn over a product for maintenance, and disband to go off to other jobs, perhaps several times a year or more.

Figures 1-1 through 1-3 show sample software development organization charts for different sized software development organizations. There are many different types of development organization structures you could come up with besides those illustrated below. Following the concepts presented, you should tailor one of these organization charts to best suit the requirements of your group.

Figure 1-1 Small Corporate Software Development Department



Project 3 Project 3
Developer 3
Project 4
Project 4 Developer 4
Project 5 Project 5 Developer 5

Figure 1-2 Medium Corporate Software Development Department

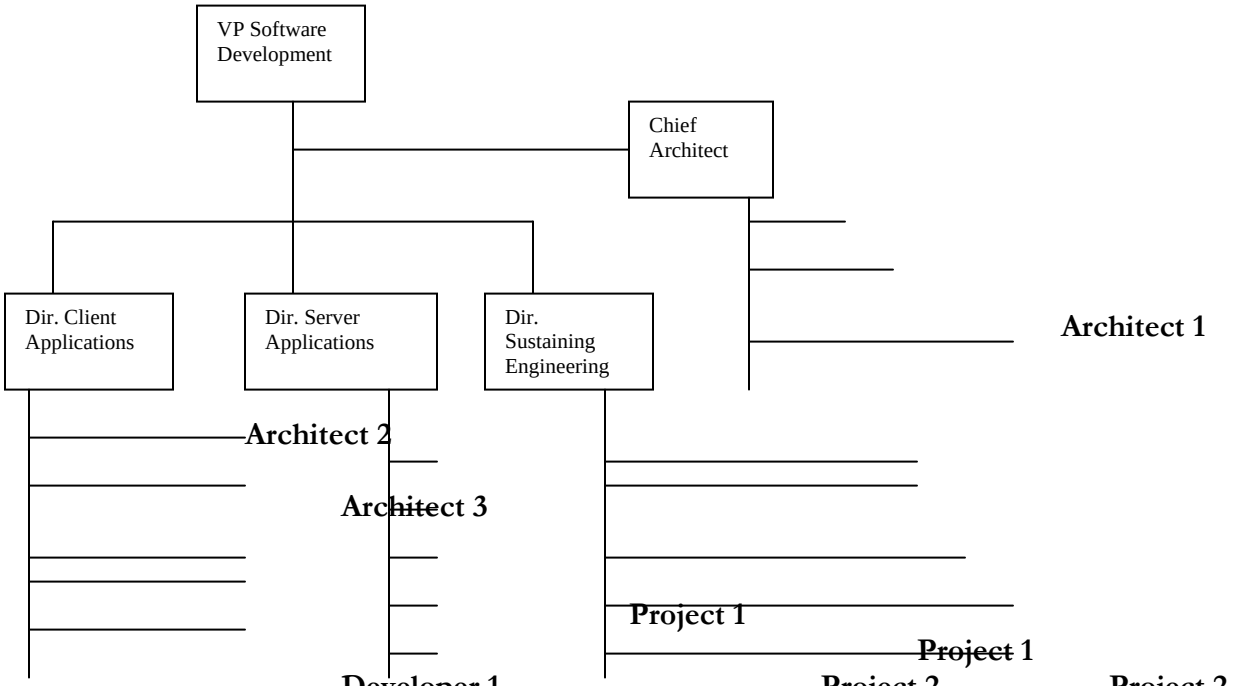


Figure 1-3 Large Corporate Software Development Department

Thirteen Organizational Structure Mistakes

No matter the size of your software development organization, there are certain mistakes you want to avoid. Many cultures consider thirteen to be an unlucky number, so we have drawn from other developers mistakes to list the thirteen organizational structures that frequently fail, no matter how lucky the manager feels. Good organizational structure is a matter of management theory, science, and experience, not luck. So here are our "unlucky thirteen" mistakes to avoid.

#1 Combining software development and operations into a single organization.

The job of operations is to keep applications up and running. The easiest way to do this is to never change anything. Combining development and operations into a single organization has the natural tendency to stifle innovation. Software development organizations should be separated from operations to allow new and modified applications to be developed as required to support the business needs of the company.

#2 Organizing software technology specialists by project.

A modern day software development project requires a wide range of software specialists during different times in the software life cycle. If you have a small organization with only one specialist and two projects, it is clear the specialist needs to work on both projects, perhaps at different times. As your organization grows and develops more specialists, you still want to have specialists work across projects wherever and whenever they are most needed. What happens if you assign technology specialists by project and each project ends up having a small number of specialists that must act as generalists while doing the work of another specialist assigned to a different project? The result your organization as a whole cannot take advantage of all its specialists where they are most needed and fails to take advantage of the synergies of being one integrated development organization.

#3 Organizing software technology specialists by application domain (i.e., financial, manufacturing, etc.).

This results in the same problem as mistake number 2, above. Given a fixed headcount, each application domain receives a smaller group of specialists who therefore are driven to become more generalists. There is no room for software technology specialists that span across application domains, for instance a GUI design specialist.

#4 Organizing software developers by delivery platform (i.e., Windows, Unix, and mainframe).

Developers tend to naturally develop biases in favor of their assigned platform. As a result, there tends to be little or no innovation in new platforms (for instance, network computers) or cross-platform approaches (for instance, web top computing or client- server systems).

#5 Separating software development and software maintenance groups.

When you separate software development and software maintenance groups, you end up requiring the same types of software specialists for each group. This means you either have to double headcount in your organization for specialists or force one group to reduce specialization. In addition, it becomes harder to create incentives for software developers to do things right the first time as they know another group will ultimately be responsible for fixing any mistakes. In addition, such an organization tends to develop two classes of software developers, leading to morale problems.

#6 Representing projects in the formal organizational structure Vs. defining project teams that cut across organizational boundaries.

This leads to a re-organization whenever a major project ends. As a result, software developers spend an inordinate amount of time looking for their next job or find unnecessary reasons to prolong their existing project.

#7 Organizing software developers into long-term and short-term development groups.

Once again, this type of organization requires two of every software specialist and builds unnecessary competition between the two groups. In addition, this organization often encourages point solutions that may be quicker to implement but cost more in the long term because of higher maintenance costs and difficulties in integrating with enterprise-wide applications.

#8 Designing organizations that need "super-developers" to succeed.

In today's complex software development environments, no single developer can be a specialist in all fields. The organization should allow and reward developers who become true specialists in a single field. Also, the workload should be managed to allow software developers to lead balanced lives. No one can work eighteen hours a day forever and be expected to maintain his or her work quality and personal satisfaction.

#9 Designing organizations that tolerate underachievers.

This is the corollary of mistake number 8. A healthy organization employs everyone's complete range of talents to their fullest. Software developers who are allowed to

underachieve will become bored with their work, which will only lead to poorer quality and further underperformance.

#10 Designing organizations that reward empire buildings.

Organizational structures should eliminate all incentives for empire building. This means providing equal career paths for both senior level software engineers and software development managers. Along the same lines, career paths should be provided both for software generalists and software specialist, as both are needed in a healthy organization.

#11 Setting organizational goals that compete against each other for customer satisfaction.

Customer satisfaction should be the ultimate goal of all software development organizations. One organization should not have goals whose achievement effects the customer satisfaction of another organization. For instance, if two development groups are working on applications for the same customer that must ultimately be integrated together, one organization should not be rewarded for meeting its timelines if this was only done at the expense of creating a more difficult integration task for the second group.

#12 Organizing around individuals Vs. personality types.

Every experienced software development manager recognizes the importance of matching a developer's job to their personality; however, even in the most stable of organizations, individuals come and go. You should not therefore, design organizations around individual personalities. Instead, organize them around more general personality types. This allows them to recognize the value of individual diversity without having to reorganize every time someone comes and goes.

#13 Mandating organizational changes from the top down.

There comes a time when all software development organizations must change to adopt to new business models, technologies, or clients. However, mandating a structural change has little or no effect if it is not accompanied by cultural and process changes. The best way to assure successful change is to manage it via a participative process where all developers are given a chance to affect the final outcome.