



Wizards, Gurus, and Saints

(Transitioning from Software Entrepreneurship to a Well Managed Firm)

Ben Lieberman, Ph.D.
Principal Architect
BioLogic Software Consulting

Introduction

Software development is a strange business. Although its practitioners would have their customers believe that the development process is based on rigorous engineering discipline, the reality is more often far from this ideal. Often the success or failure of a development effort hinges on the talents and support of a few key individuals in an organization - usually the highest paid and most highly sought after individuals. These "indispensable" people seem to fall into three broad categories: Wizards, Gurus, and Saints. We will begin by defining what is meant by these terms, discuss how and why they come into existence, and then note where in the life of a company a transition must be made from entrepreneurial efforts to a well-managed firm.

Identify the Roles

A Wizard is the individual on a project who is aware of all the magical incantations that are necessary to get a particular section of the system to operate properly. Usually these are developers who first wrote the code for the section of software in question. They may also represent system engineers who understand the physical configuration of the system. No one would think of working on this particular part of the system without consultation with the Wizard, who after all knows the code better than anyone else. These folks are the mover-shakers who can get a project done and shipped and money rolling into the company. Wizards are typically very talented, driven people whose desire to complete a project can assist the entire team in overcoming many initial obstacles.

While there is no question that many projects have died on the vine due to slow market emergence, once a product has moved into the maintenance phase (even if it is referred to as a minor release) there is no excuse, and a great deal of benefit to be gained by, reducing the dependency on these individuals. Although, as a rule, these are highly motivated and quite loyal people, wizards have the notorious reputation of performing their magic for the highest bidder. Clearly, once the Wizard has walked out the door, so has most of the understanding of the system structure.

The second key individual in an emerging organization is the Guru. These people have the understanding of why the software was created in the first place. Often this information is gained from the customer, but just as often it is introduced as development progresses and problems are encountered and addressed. Just as a Wizard would always be consulted on a matter of code, the Guru would always be consulted on a design decision. The wisdom of the company, normally represented by the requirement base presented to development, is again captured primarily in the head of one or a few core individuals. Gurus are most often the System Analysts who meet with the customer, but may also be represented by developers, Project Managers, and even sales staff.

There is little time in an initial development effort to record all of the changes introduced by the customer. In fact, often the only way to capture all of these changes is to hold them in your head, which is the most flexible development tool in the software industry. Unfortunately, it is also the most difficult to control and access. Thus, similarly to the Wizards, the Guru will be a sought after commodity by many disciples. Since much if not all of the system understanding is captured in the Guru's head, when they are promoted, resign, or are hit by a large moving van there is little opportunity for anyone else to take over the Guru's role.

Finally, we have the Saint. These are clearly necessary for any initial company to survive and will often times come bundled with either a Wizard or Guru. Simply enough, a Saint is someone who is willing to sacrifice for the good of the company. They will work long hours, step up for heroic efforts when the software must work on schedule, and rally the team to greater achievement when the chips are down. Saints have been well known to be the primary motivator in miraculous recoveries.

The main drawback to a Saint is that they are human. Humans are imperfect beings (even if a development Wizard would have you think otherwise) who have rather severe limits. If you increase the workload on an individual beyond their capacity, there will be a dramatic, and often disastrous, drop in both productivity and accuracy. There will always be emergency situations that cannot be foreseen, but heavy reliance on a few key Saints is a dangerous way to run a business; Saints have a nasty habit of becoming Martyrs.

All of these individuals seem to play a vital role in the survival of the initial project, so why would you want to avoid them? Isn't the name of the game product to market? Yes and no. In the short term, survival is key and a top flight Wizard, Guru, or Saint (WGS) will be necessary to success. In the early days of any entrepreneurial venture, the benefits of flexibility, motivation and unbounded innovation are crucial to initial success. It is only during the transition to a larger, maintainable system that this approach falters and ultimately fails (see (Royce 1999), pg 211). The culprit is again the fact that these folks are human. In addition to the aforementioned workload limits, humans also have a limited capacity to hold highly complex and detailed schema in their heads for extended periods. If the company has long-term survival in mind, with plans to maintain the software beyond initial release, then it will inevitably become increasingly complex. At a certain point it will overwhelm even the most talented Wizard, bewilder the wisest Guru, and martyr the most dedicated Saint. So how can we transition from a highly flexible but limited state, to a possibly more rigid but stable process?

Recognize Late Status Quo

It is not a simple process for a company to recognize and begin the transition process from entrepreneurship to a well-managed firm. The cost of either beginning too soon or too late is equally high, and potentially fatal. It is therefore critical to develop ways to recognize the initial signs of difficulty and create ways to reduce the pain involved in reworking the firm.

Eric Flamholtz of the UCLA Anderson School of Management (Flamholtz 1990) notes that at a point roughly between the range of \$300,000 and \$3 million in total revenue for a service firm, such as software development, that the company will begin to experience significant "growing pains." These are evidenced both in the behavior of the individuals that compose the firm, as well as the quality of the product and ability to deliver on schedule. It is key at this point that the firm begins to develop a "professionalized" working environment. This includes the creation/modification of the supporting corporate organization structure, the operational processes, the perceived/actual roles of the staff, and the culture of the firm. The remainder of this article will focus on the staff related issues.

Some of the initial warning signs of stretching staff resources are an increasing workload on your primary WGS personnel. There will be increasing grumbling of "I am way overloaded right now" and "I am too busy to be bothered with that". Important decisions are rushed or deferred in preference to perceived critical delivery, with equally disastrous results! Fire-fighting and crisis management becomes the normal daily behavior.

The typical response from a corporate level is to throw people at the problem - hire a large group of consultants or new employees to supplement the WGS's efforts. This very expensive solution will, unfortunately, have little or no positive impact on the WGS workload since, in the absence of system documentation, they are the only ones who know how things work (Weinberg 1992). Moreover, it will take a great deal of time for the new employees to reach the vaunted status of WGS, again primarily because there are no written materials from which to learn. In fact, the workload on the WGS will double since they now have to mentor the new "apprentices" in addition to completing all of their original work.

A second sign of impending doom is a marked decrease in worker productivity and accuracy. This usually manifests itself as a dramatic increase in the number and severity of system level errors, leading to an overall decrease in product stability and quality. As noted above, the complexity of the project will continue to grow as the number of customers and users grows. This complexity will eventually surpass the ability of any one individual, no matter how gifted, to maintain within themselves. Since the entire picture of the system dependencies is now unattainable, errors of omission and ignorance will naturally occur - resulting in a fragile, non-maintainable system. More insidiously, the WGS may continue to believe that they are still in command of understanding the system, and thus resist the necessary changes that need to be implemented to prevent complete collapse. As "ship now - fix later" becomes the operational process costs will begin to soar. To paraphrase Philip Crosby (Crosby 1996), quality is cheap - mistakes are expensive; it will always cost more to repair than to avoid error in the first place.

The final sign of the apocalypse is the loss of the WGS themselves. If no attempt is made to capture the knowledge of the WGS prior to this point, they will understandably leave for greener pastures, taking all of that hard won wisdom with them. Although this can be quite a boon for the competition, it does little for the company left behind. The remaining staff is then stretched to cover the areas that the WGS had previously been working on, with the additional burdens of:

1. Mentoring new staff (introduced to replace the WGS),
2. Deciphering the poorly or undocumented legacy system, and
3. Completing all of their original work.

At this point collapse is all but inevitable. A new form of culture needs to be established, one that is less reliant on the heroic efforts of a few talented people and can leverage the greater pool of talent in the organization. This begins by altering the roles played by the staff.

Transitioning Roles

So far, I have discussed the initial roles of the WGS, some of the consequences of ignoring this state for too long, and some of the warning signs to watch out for. Now I will discuss the transition roles that correspond to the Wizard, Guru, or Saint.

As noted above a Wizard captures within themselves the magical incantations hidden deep within the fabric of the code universe. As the project stabilizes a Wizard needs to become an *Engineer*. An Engineer works best with a defined path and established, best practice procedures. There is still a great deal of room for innovation, but not the trailblazing type of work expected from the Wizard. Some of the costs and benefits of this transition are:

- + Establishment of a uniform process
- + Incremental improvement
- + Easier transmission of process and system structure information to new employees

- Slower, more deliberate pace of development
- Restricted innovation opportunities
- Multiple information sources must be accessed to collect the full system picture

Since a Guru represents the wisdom of the system and often interacts directly with the customer of the product, the natural transition type is the *Architect*. This transition is represented by the need to capture and publish the system level knowledge. The complexity of the system has exceeded any single Guru's abilities and so gaps have begun to appear in the tapestry of the system. These areas have the highest level of "gottcha". Architects are best at capturing the problem defined by the customer and presenting it in a format understood by the Engineers, and vice-versa. While the Engineer sees a tree, the Architect sees the grove. This broad view loses some of the detail while gaining the ability to calculate the impact on changes to the

entire system that is more formal than the intuitive guesses provided by the Guru. Some of the costs and benefits of this transition are:

- + Creation of a visual representation of the system
- + A single unified description of the system
- + Creation of a communications center for all development efforts (analysis, code, test)

- A more deliberate development pace
- High level focus may neglect critical details
- ± Process improvement innovators (Chaos generators)

The final entry above is a plus and minus since as innovators Architects can introduce useful process improvements, but these introductions are usually foreign elements in themselves that can lead to initial decreases in productivity.

The final transition to discuss is the Saint to a *Manager*. At some point the workload required to keep the system alive surpasses a single humans ability to go without sleep. Since a 25 hour workday is a physical impossibility in our space-time continuum, it is necessary to distribute the work. This work must be organized and delegated to a team of people for execution. The primary role of the manager is to enable all of the team members to recognize the scope of the problem, rather than to assign tasks on small parts of the problem. Since the goal here is to avoid concentration of effort into the hands of one person, all that is accomplished with the assignment of tasks is to force the manager to merge all of the efforts. In many ways, a Manager is more like an editor; they set the overall goal and then work with the team to establish the objectives and to collect the results for presentation. Some of the major benefits and risks associated with a Manager are:

- + Division of Labor
- + Interplay of ideas between team members
- + Synergistic team efforts

- Potential for personal conflicts
- Bottlenecks due to differing productivity levels
- Additional project planning overhead

Since the overall goal of this effort is to increase the "Corporate Memory," an emphasis must be placed on the increased cost and time devoted to the recording process. This can be as simple as recording meeting notes, or as complex as creating a full corporate library of system characteristics.

Conclusion

Although this paper has tried to simplify the software development world into three categories, it is obviously far more complex. However, the need to document, model, and distribute information about the company's problem domain is a growing reality in our industry. By looking at the roles and responsibilities of each member of the organization, it will be possible to alter and grow those roles to better suit the increasingly complex world of software development. As I am fond of saying, "If you are taking two steps back for each step forward, try turning around."

References

Crosby, P. (1996). Quality is Still Free. New York, McGraw-Hill.

Flamholtz, E. (1990). Growing Pains - How to Make the Transition from an Entrepreneurship to a Professionally Managed Firm. San Francisco, Jossey-Bass Publications.

Royce, W. (1999). Software Project Management, A Unified Framework. Reading, MA, Addison Wesley Longman, Inc.

Weinberg, G. M. (1992). Quality Software Management: Volume I, System Thinking. New York, Dorset House Publishing.







Initial Role	Transformed Role	Positive Aspects	Negative Aspects
Wizard 	Engineer 	<ul style="list-style-type: none"> • Establishment of a uniform process • Incremental improvement • Easier transmission of information 	<ul style="list-style-type: none"> • Slower, more deliberate pace • Restricted innovation opportunity • Multiple information sources to obtain full system picture
Guru 	Architect 	<ul style="list-style-type: none"> • Visual representation of the system • Unified system view • Communication center point for all system information/models 	<ul style="list-style-type: none"> • Slower, more deliberate pace • High level focus may neglect critical details • Process change innovators (chaos)
Saint 	Manager 	<ul style="list-style-type: none"> • Division of labor • Interplay of ideas • Synergistic team efforts • Able to handle greater complexity of problems 	<ul style="list-style-type: none"> • Increased potential for interpersonal conflicts • Bottleneck due to different productivity levels • Project planning overhead

Table 1. Transition of Roles: benefits and problems.