The Longest Yard: Reorganizing IT for Success

By Ruby Raley and Bruce F. Webster

[Originally published in the September 2006 issue of the Cutter IT Journal*; this version has some minor edits. This document may be freely redistributed but may not be sold or reproduced in physical publications without express permission of the authors.]*

Introduction

Managing talent and skilled workers is a key success factor in the 21st century across all industries, not just technology-based industries. However, many of our organizational practices originated in the 1800s with railroading and the rise of the Industrial Age. Your average Gen Y programmer bears little resemblance to the unskilled and semiskilled laborers of the past. Just as technology has evolved, so must our organizational practices.

We have observed that firms (corporations, government agencies, etc.) with consistently successful IT organizations share certain traits with sports teams — specifically, American football teams. This article looks at what makes a successful football team and then applies those traits to IT organizations.

Personnel

Software engineering studies repeatedly point out that the people involved represent the single most significant success factor in an IT project [2, 3, 8], yet most firms treat IT personnel as though they were in a “parts from the lowest bidder” situation. Many firms struggle with their IT organization because of lack of attention to personnel issues — in particular, recruiting IT personnel, assigning them to appropriate positions, and coaching them for best performance. Most firms do not ensure strong relationships and goal alignment among IT and business stakeholders.

Recruiting

Numerous studies in software engineering have shown wide variations in productivity and talent among IT personnel, even when education and experience are equivalent [8, 11]. Other studies have indicated that smaller IT teams tend to do better than larger IT teams, due to issues such as communication [4]. Still other writings point out the strong need for teamwork and cooperation within IT development groups [6, 7].

Unfortunately, most firms do not recruit IT personnel with those factors in mind. Instead, they focus on checkbox items, such as certifications and years of (claimed) experience with a given technology. They seek to match keywords against what they think their needs are. They conduct at most a few interviews with a given candidate — typically by managers, with at most one technical interview. They often fail to evaluate candidates by talent, skill, attitude, and ability to cooperate — by and large, they do not know how. And they tend to believe, unconsciously or explicitly, in the mythical man-month, thinking that the larger their IT staff, the quicker they can complete projects [5].

By contrast, sports teams invest significant time and resources in recruiting a small number of players. They read the sport press, review scouting reports, watch game films, analyze performance statistics, conduct interviews and physical examinations, and may require a tryout before agreeing to take on a given player. They may have a probation period during which the player has to either make the team or be let go. They recognize that they have a limited number of slots on the team, and they may also be operating under financial constraints, either due to the team's own resources or due to a league-imposed salary cap. They then negotiate a contract commensurate with the player's talent and skills, physical condition, and performance to date, as well as their own need to fill a given position, any financial restrictions they may face, and competing offers from other teams.

How would this look in an IT organization? Here are some ideas:

- **Recruit based on reputation and recommendation**. We have found that the single best predictor of performance of an IT engineer is a strong recommendation from an IT engineer we already trust and respect. If you don't have an existing recommendation, then have all your best IT people interview the candidate.
- **Focus on talent and professionalism**. Talent is critical, often more so than specific technology experience. A talented IT engineer can pick up new technologies rapidly, but an untalented IT engineer, however experienced, will be a drag on the project. At the same time, having talent is no excuse for being a prima donna. We'd rather have someone who is a bit less talented but gets the work done. Unfortunately, we lack strong metrics for predicting talent. This is why the recruiting process is so important, as we discuss below.
- **Work with a team size limit and a salary cap**. It is our observation that firms often have too many IT engineers rather than too few. This is often a direct result of management policies and politics that equate headcount with internal clout — in short, kingdom building. That can be a hard hurdle to overcome, particularly since such politics are often entrenched in the firm.

At Pages Software (a venture-funded startup where Bruce was CTO), we applied these techniques to build our engineering team from scratch. All Pages engineers made recommendations as to which candidates to bring in for interviews, either from incoming résumés or from their own professional network. When a candidate was selected to come in for interviews, every Pages engineer would conduct his or her own one-on-one interview with that candidate, each of us using our own preferred style. We would then meet to evaluate how that person would meet our current and future needs and how well that person would fit into our team. We only had a certain number of slots, and we were going to be working very long hours together for years, so we chose carefully. The result was a very stable and talented development team that had almost no turnover over a grueling four-year period.

Recruiting is extremely important, so staff the function carefully and be sure to work closely with your new team members so that they can become productive quickly.

Organizing the Team

Again, many studies have recommended ways IT teams should be organized, starting as far back as 1969 [1] up to the latest debates surrounding agile development [10]. However, in focusing on organization, we often lose sight of what it takes to build a team.

A Quarterback

The quarterback is the most visible player on a football team and is responsible for leading the offensive team to get the ball into the end zone. The equivalent position on a software team is the chief architect. Fred Brooks first laid out the need for a chief architect at length, most particularly for conceptual unity [5]. We believe that his observation stands but that it is too often ignored. For example, one project we consulted on had 30 engineers developing network management software for a global telecommunications project — but no architect (or architecture) at all! Diagrams existed for half a dozen subsystems, but those diagrams did not connect up with each other. One of our first tasks was to create an architecture for the project, then adjust the subsystems to fit within that architecture.

Mutually Supporting Team Members with a Common Goal

This may sound obvious, but think: how many of you can say this describes your IT department? Imagine that our hypothetical football team has just scored, and the receiver who caught the ball is doing a victory dance for the camera and the local fans. Does the culture at your firm reward “displays of unsportsmanlike conduct” or penalize that behavior? There are IT shops where a flourishing hero culture exists. Management rewards those individuals almost exclusively, while their fellow team members smart from their jibes and never get to carry the ball.

We remember a release cycle at one firm that illustrates how teams can be torn apart. The senior architect delivered a new release of middleware to the product team with no documentation and no test scenarios. A junior QA engineer started the validation of the middleware, working day and night to test and document the system. After days of work and unanswered pleas for help, he sent his team lead and the architect an e-mail stating, “It doesn’t work,” and went home for the night. The architect’s response? He sent a note to the entire team, including the CTO, accusing the junior team member of leaving the ball on the field and insinuating that his incompetence was the real issue. The result? The release never delivered a performance improvement, team communications suffered for months, yet the architect was rewarded and touted by management.

Successful IT organizations need clear technical leadership at multiple levels that is supportive and makes strong contributions to team success.

Coaching

Once you have what you consider to be the right personnel in place, you still need to coach them. Professional sports teams recruit the best of the best, pay them millions of dollars, and yet still focus significant amounts of time and money on guiding these top-notch athletes to superior performance through coaching and the use of playbooks. Why is it that we think we can hire IT professionals off the street, throw them into cubicles, and then expect them to work as a team to achieve the firm’s business goals via technology?

Coaching Staff

A professional American football team typically has a head coach with several specialty coaches (QA, backfield, receivers, offensive line, defensive line, etc.). The head coach builds and oversees the entire team, interacts with the business side of the firm, devises how the team is going to win the championship, makes position assignments, adds or cuts players, and motivates, encourages, and/or excoriates players as appropriate. The specialty coaches work with team members in their specific responsibilities to develop requisite skills.

Each IT department and project head needs to see him- or herself as a coach. A coach is held accountable for the overall success of the team and is focused on improving both individual and team performance. Likewise, if you’re in IT management, you should focus first on building the right team for the project at hand, then coaching each IT engineer in his or her responsibility. If an individual isn’t performing and doesn’t improve under direct coaching, then find someone who will perform. Frankly, this can be tough; some very talented IT engineers, like some very talented athletes, can become prima donnas, and one of the hardest decisions you may face is telling such a person to either work as part of the team or leave it (and possibly the firm).

“Specialty coaches” would correspond to mentors, an idea much touted in management circles but seldom applied to the IT organization. At OSG Incorporated, we have seen the effectiveness of placing a handful of seasoned technical mentors — specialized in areas such as analysis, architecture, design, implementation, and quality, as well as key technologies — within large IT organizations. Mentors work with the IT engineers to solve problems, improve performance, and generally convey the hard-won experience and insight that only come from years in the trenches.

Philosophy and Playbooks

Successful head coaches are usually known for having a definite philosophy, which is reflected in the assistant coaches they hire, the players they recruit, the offenses and defenses they implement, and their approaches to player coaching and discipline, team practice, play-calling, and game strategy. Similarly, each football team has its own playbook, which contains both offensive and defensive sets that reflect the head coach’s philosophy and/or the expertise of the assistant coaches, as well as the particular strengths and weaknesses of the players themselves. The team practices these plays and sets until they become second nature; the plays and sets are then called as appropriate by the coaches, quarterbacks, and/or defensive team captains.

It’s important to note that successful head coaches don’t panic or throw out the playbook when things go wrong. They may make a few substitutions or change the offensive play/defensive set mix, but they tend to stick with their philosophy and keep doing those things that made them successful in the first place.

While much as been written, starting largely with Brooks [5], about the need for conceptual unity of system architecture, less has been written, said or done about conceptual unity across the entire IT department — the equivalent of an IT coaching philosophy. In our experience, few IT organizations apply a consistent IT philosophy to such diverse issues as software methodologies, enterprise-wide technical architectures, software architecture and design patterns, coding standards, development tools, reusable deliverables, QA processes and tools, and component- and service-oriented architectures (not to mention recruiting, training, and assigning IT personnel).

As a result, we have observed significant mismatches, conflicts, and gaps among those aspects of specific IT organizations. For example, we have consulted at a firm that sought to follow a highly structured formal methodology while at the same time implementing a “weekly software build” schedule. The result was that, strictly speaking, developers only had about eight hours to do coding each week, since the rest of their time was taken up by paperwork and formal design, implementation, and testing reviews.

By contrast, we have worked with senior IT executives who had a set of explicit rules and principles that they typed up, handed out, and would cite when decisions and issues came up regarding the various processes, tasks, and technologies cited above. Indeed, we have done this ourselves on occasion — we have found this prevents many problems and simplifies many others. A good template is the Agile Manifesto and the associated "Principles behind the Agile Manifesto" [10]. We say this not because we unconditionally endorse the agile approach, but because the Agile Manifesto and Principles together form an outstanding example of a self-consistent IT "coaching philosophy," all expressed in fewer than 20 statements. Another source that we both have used professionally is the list of 180 or so "heuristics for systems-level architecting" collected by Mark Maier and Eberhardt Rechtin [9].

The IT equivalent of a playbook would start with this written philosophy, explaining the principles. It would then build upon the processes, tasks, and technologies listed above, explaining how (within this particular IT organization) you work with your teammates to carry out specific key IT development tasks or solve particular problems. Even we aren't sure just what such a playbook would look like — other than it would have to be relatively small, simple to look through and read, and easy to update.

Performance

Having recruited the right personnel and taught them the plays, the coach now has to get the team to go out and perform — do the blocking, passing, running, and tackling necessary to win the game.

We picked American football as our major analogy for several reasons; most important is the relative balance between offense and defense, which we correlate to development and quality assurance, respectively. (The other software development lifecycle activities can, for now, be relegated to either "coaching" or "special teams.")

Offense = Development

The development team is essential to winning the game: delivering working code that meets requirements on time. And just as different football teams use different offensive philosophies, your IT organization needs to decide what offense each IT team will use.

West Coast Offense (Adaptive)

A passing offense inherently takes a greater risk in each play. There is the opportunity for interceptions and missed passes, but there is also the opportunity for fast scores and game-changing plays. The IT equivalent is an "adaptive" IT methodology, such as the various agile approaches. This makes sense when business drivers require rapid development and close interaction with end users, such as when business drivers change on a rapid basis (due to competition) and a fast, tight development cycle is required. For example, OSG has developed a technology for data extraction and visualization that can be implemented, deployed, and customized in a matter of weeks in large organizations with established IT infrastructures. We use an agile methodology because the payoff for clients comes from a very rapid turnaround time, both in getting the data *now* and in implementing custom visualizations in short order.

East Coast Offense (Predictive)

A running offense seeks to control the ball, grind away at the opposing team, and occasionally break loose for long yardage by focusing the best running back on the weak areas of the defense. The IT equivalent is a “predictive” or formal methodology, such as a classic or modified waterfall approach. This makes sense when business drivers and technological considerations require a significant investment in analysis, architecture, and design up front; for example, in industries where reliability is an issue and the system lifecycle may be five to 10 years. We have seen the successful application of such an approach in custom development of an enterprise-wide IT system for a utility firm, with formal signoff not just of deliverables but also of the acceptance criteria for those deliverables.

Balanced Offense (Iterative)

A balanced offense alternates evenly between passing and running, picking the best play as the situation demands. The IT equivalent is a modern iterative methodology, such as the Rational Unified Process (RUP). This makes sense when you need to make some up-front investment in analysis and architecture due to the scale and complexity of the systems under development, but you don’t want to wait months or years for initial deployment; for example, when re-engineering and replacing legacy systems.

Just as in football, there are fierce proponents and critics of each of the IT approaches above. The key is to apply them on a case-by-case, team-by-team basis — and to be willing to change or adapt if the selected approach isn’t working well.

Defense = Quality Assurance

Football teams that only excel at offense usually play exciting games and often win individual games, but they seldom end up as champions. That role is reserved for teams that excel at defense as well. The same is true for IT organizations. They may have great developers and flashy deliveries, but without a solid quality assurance (QA) team, they seldom ship a completed system on time, if at all. In our experience, most IT organizations are understaffed in QA. The QA department is viewed as a checkbox in the software lifecycle and is given little recognition, management attention, or budget. *Many IT organizations would likely improve their IT project success rate by doubling their QA staff and halving their development staff.*

Quality assurance is more than testing, although most organizations tend to equate the two. Instead, QA include:

- Ensuring appropriate expertise (subject matter, technological, methodological)
- Publishing guidelines and standards for every aspect of the software development lifecycle
- Gathering and using metrics
- Conducting reviews
- Carrying out a full spectrum of tests
- Managing defects and deliverables
- Having a formal software release process
- Collecting feedback for future improvements

The key word for all these activities is “appropriate” — appropriate expertise, appropriate guidelines, appropriate metrics, and so on. And while the “offense” has the task of delivering working code, the “defense” — the QA staff — is responsible for ensuring the code achieve acceptable reliability, performance, and functionality before it goes into production.

Once again, we can draw upon a few general football defense concepts for our IT equivalents:

Blitz Defense (Adaptive)

In this defense, most defensive players rush into the backfield to tackle the quarterback (or running back). This reflects agile concepts such as test-before-code and pair programming [10], as well as more general concepts such as unit, class, and module testing. Tests are defined before the code is written and are implemented at the lowest code unit levels (procedure, class, etc.). Programmers work in pairs to provide immediate feedback on design and implementation decisions as the code and corresponding tests are written. Likewise, this defense encompasses such QA concepts as coding/design standards and expertise of both developers and subject matter experts. The goal is to ensure zero defects before the code is ever released to higher-level integration and testing.

Man-to-Man Defense (Adaptive/Predictive)

In this defense, most defenders have a specific back or receiver to cover; each follows his assigned player wherever he goes. This reflects integration, interface, and end-to-end testing, as well as configuration management tools and practices. The goal is to ensure that complete scenarios and uses cases can be carried out on individual applications and virtual applications (collections of applications that work together to carry out a given use case).

Zone Defense (Predictive/Adaptive)

In this defense, most defenders have a certain area (zone) on the field to defend; each covers any runner or receiver who enters that zone. This reflects category-based testing, such as performance and stress testing, computational verification, scheduling testing, and regulatory testing, along with classic defect and change control management. The goal is to ensure that individual and virtual applications meet certain system and business requirements.

Prevent Defense (Predictive)

In this defense, most defenders drop back from the line of scrimmage in order to prevent a successful long pass play. This reflects classic waterfall concepts such as delaying most testing until coding is complete; conducting whole-system tests such as compatibility/parallel testing, restart recovery testing, user acceptance testing, and production readiness testing; and performing release management (alpha, beta, production releases). The goal is to ensure that all stakeholders agree that it is both safe and desirable for the new system to go into production.

The important point is this: QA is every bit as critical to winning as development and should have equivalent resources, including personnel and preparation.

Dealing with the Front Office

Professional football teams are a business. Their purpose, beyond the ego gratification of the owners, is to make money. Usually this requires performing well enough to sell lots of tickets, merchandise, broadcast rights, sponsorship rights, and so on. That money, in turn, provides training facilities, equipment, salaries, and benefits for the players. Note that spending lots of money, even with a well-respected head coach, does not necessarily guarantee victories and championships. Witness, for example, the travails of the Washington Redskins over the past several years (although, by all accounts, the franchise itself remains very profitable).

Likewise, an IT organization depends upon the business side of the firm for its funding. The IT organization needs to “win” by meeting certain key business drivers, thus encouraging the business side to continue to make those funds available to the IT organization. Note, also, that spending lots of money on IT does not necessarily guarantee successful delivery of the intended system, much less the expected return on investment (ROI). We have personally reviewed failed IT projects costing over half a billion dollars and troubled IT projects costing well over a billion dollars.

The bad news is that, unlike in football, the business and IT sides of a firm don’t always agree on what constitutes a “victory” (even though both sides can usually agree on what a “loss” is, at least in cases of total or significant project failure). Indeed, sometimes they cannot fully agree one what the *game* is.

Typically, the business side has three overlapping goals for IT:

- B1 — provide required and sufficient functionality to allow the firm to operate and compete on a level playing field
- B2 — provide superior or unique functionality to allow the firm to beat its competition
- B3 — provide efficiencies in productivity to allow the firm to free up funds for investment, expansion, and/or profits

Likewise, the IT side typically has three overlapping goals for itself:

- IT1 — maintain its existing systems (and staff) or their equivalent
- IT2 — migrate off aging, obsolete, or defective technology
- IT3 — keep the business side (including end users) happy, or at least off its collective back, while getting the funds necessary to accomplish IT1 and IT2

These goals overlap and coincide to a certain extent, but there are some dangerous and often unexamined gaps in there as well. For example, say the business side proposes a project for goals B2 and/or B3. IT will see this project as a chance to meeting goal IT2 (and, with luck, IT1 and IT3) and will design accordingly. IT delivers the project into production on time and on budget, feeling that it was succeeded on all counts. However, the business side finds that the time elapsed has shifted most of the B2 (competitive) benefits over to B1 (required/sufficient), since the competition has likewise progressed. Furthermore, the increases in IT maintenance costs (due to some level of legacy system carryover) and the inefficiencies due to the inevitable disruption caused by changes to previous business processes have reduced, delayed, or eliminated the hoped-for B3 benefits.

Now that’s what can happen when IT actually does understand the business side’s requirements *and* delivers the system on time and on budget. Imagine the frustration and disconnect when (as is often the case) the IT and business sides cannot effectively agree upon project scope and requirements, core business drivers, or key performance indications — *and* the IT project is late and over budget. No wonder authors from Paul Strassman to Nicholas Carr have seriously questioned the competitive advantage of IT, in essence arguing that B1 is the only goal the business side can hope for.

We believe that B2 and B3 goals can be achieved — but that it will be harder and less rewarding unless firms rethink, perhaps radically, their entire approach to IT development and deployment along the lines we have laid out in this article.

The End Zone

We believe you will find value in thinking of your IT organization in terms of a competitive sports team and implementing some of the suggestions above. We further believe that in doing so you can improve your project success rate and increase employee retention. These concepts don't necessarily require wholesale revamping of your IT organization; many can be implemented piecemeal or in phases, which makes them all the more attractive. The idea is to rethink your IT organization by recruiting top-notch personnel, preparing them for success, molding them into a team, and then helping them to perform in order to achieve your firm's critical business goals.

References

[1] Aron, J. D. "The 'Super-Programmer Project.'" Reprinted in *Classics in Software Engineering*, edited by Edward Nash Yourdon. Yourdon Press, 1979.

[2] Boehm, Barry W., Maria H. Penedo, E. Don Stuckle, Robert D. Williams, and Arthur B. Pyster. "A Software Development Environment for Improving Productivity." Reprinted in *Software State of the Art*, edited by Tom DeMarco and Timothy Lister. Dorset House, 1990.

[3] Booch, Grady. *Object Solutions*. Addison-Wesley, 1996.

[4] Briand, Lionel C., Khaled El Emam, and Isabella Wieczorek. "Explaining the Cost of European Space and Military Projects." In *Proceedings of the 1999 International Conference on Software Engineering*. IEEE, 1999, pp. 303-312.

[5] Brooks, Frederick P., Jr. *The Mythical Man-Month*. Addison-Wesley, 1975.

[6] DeMarco, Tom, and Timothy Lister. *Peopleware*. Dorset House, 1987.

[7] Hohmann, Luke. *Journey of the Software Professional*. Prentice Hall PTR, 1997.

[8] Keyes, Jessica. *Software Engineering Handbook*. Auerbach Publications, 2003.

[9] Maier, Mark, and Eberhardt Rechtin. *The Art of Systems Architecting*. 2nd ed. CRC Press, 2002.

[10] Martin, Robert C. *Agile Software Development*. Prentice Hall, 2003.

[11] Nash, Sarah H., and Samuel T. Redwine. “People and Organizations in Software Production: A Review of the Literature.” *ACM SIGCPR Computer Personnel*, Vol. 11, No. 3, January 1988, pp. 10-21.