Power Developer Squads Model

Resource Management Model for In-house Business Application Development Projects

Denis Urusov
Bear, Stearns and Co. Inc

Overview

It is no secret that software development projects fail more often than we'd want them to. I do not want to throw the statistics in (they are not pretty), but we all know very well that way too many projects go beyond the planned end dates and allocated budgets. One of the most cited reasons for the high failure rate is sadly, poor project management.

Of course, there are best practices promoted by Project Management Institute (PMI), Software Engineering Institute (SEI) and other organizations, but there is obviously a room for improvement. This especially true for IT project management and in this article I will address one aspect of it, namely – principle of organizing development team(s) to work on a project.

More often than not, when allocating resources to work on a new project, Project Managers simply bunch together whoever is available at the moment, without paying much attention to the actual skills that the project will require. Sometimes this is the result of an existing resource constraint, sometimes – pure negligence. This approach causes unnecessary delays (time spent in learning curves), low quality code and deployment problems. The problem is acknowledged and there are a few widely accepted guidelines as to how development teams should be built, plus using common sense is often a valid choice.

Power Developer Squads

In this article I will present a resource management model that could be adapted for creating development teams set to take on business application development projects. The model is dubbed:

"**Power Developer Squads (PDS) model**"

It is based on the following basic concepts:

- Power developer – a new role for a business application developer. In addition to the traditional function of writing code, the "power" developer assumes responsibility for at least one extra function, such as business analysis, architecting or document writing.
- Squads – minimal size development teams composed (primarily) of power developers.

These concepts are described in detail further below.

Model scope

The scope of the model applicability is preliminary limited to in-house business application development environments, which are very common for Wall Street

banks, brokerage companies and law firms. Such environments, as opposed to the ones that exist in software vendor companies, are unique in a sense that they require presence and constant interaction of all the "ingredients" needed for a business application to be produced - business users, operations and security departments, development team and QA group. This combination forces developers to participate in all the relevant phases of SDLC - requirements gathering and definition, architecture and design, implementation, systems and QA testing, deployment and release.

Power Developer - Developer redefined

As I plan to show here, modern business application developers must to be more than just code writers. Why? Simply because raising a developer's responsibilities above pure coding undeniably makes business application development process more efficient and surely benefits the entire organization in the long run.

In taking a business application through its life cycle, developers should be involved in various activities, such as working with business users on definition of business and functional requirements, writing technical documentation, making sure that the application fits into bigger infrastructure and giving operational folks a hand in the application deployment process.

Developers can and often do make invaluable contributions of their technology expertise and application domain knowledge to guarantee the success of all the SDLC phases. So we here redefine the traditional meaning of what being a developer means, and introduce a notion of a "Power Developer" – someone who is skilful in programming and plays one or more SLDC-implied roles:

 Power Developer **=** **Programmer role** +	**Business Analyst role** (interact with business users) **Documentation Writer role** (help prepare requirements and technical documentation) **Administrator role** (interact with SA, DBA and security) **Architect role** (oversee application's architecture and design, interact with R&D)

Does a power developer have to be able to do it all? Of course not, but the more roles he or she takes on – the better. The additional skills acquired by developers will advance the communication processes that are essential in SDLC. Let us take a close look at what exactly would a developer's new role call for:

Programmer - Business Analyst role

A developer that directly communicates with the business users will have a much better understanding of what the users actually want and there will be less of a broken telephone game. Traditional Business Analysts and liaisons often lack knowledge of underlying technology, meaning that they are not able to spot caveats early. They are not able to offer a better or more cost-efficient technology solution to their business users.

It seems that the industry has somewhat recognized this problem and it is now common to see a few senior-level Developers at requirements gathering discussions. Playing this the role would require excellent communication skills and be willing to understand the business needs from a developer perspective.

Programmer - Documentation Writer role

Turning to the documentation writing problem, it is worth noticing that in-house development, due to its nature, does not press application developers to keep documentation updated. Their products are not to be sold outside, the application. Maintenance and support is most often done by the same people who wrote the code – hence the popular tendency to rely on one's memory and scattered notes.

However, for highly-efficient companies that are really trying to keep the IT standards up, maintaining project documentation is vital. Having a good writer in your team increases your chances of success but not all of developers are up to the task. Writing, technical or not, is as good a skill as anything else, plus it can get really detailed and boring – a big turn-off for creative minds. So the choice is either to hire a writer from outside or find a talent in-house.

It is easy to see that no one knows the application development, infrastructure and deployment process better than the one who's working on it on a daily basis. From this point of view, it makes sense to make sure that one or more members of your team are ready to take on the documentation. If hiring a writer from outside is a company policy, you should still have someone who could competently cooperate with the professional writer, supplying the latter with drafts, schemas and engineering notes.

Programmer - Administrator role

The process of deploying an application is often overlooked. Modern development teams tend to focus on meeting the functional requirements, and leaving the deployment considerations to the last minute. In complex multi-tiered infrastructures this pattern straightforwardly leads to unexpected delays and slipping deadlines. It is important to make sure that system and database administrators are kept in the loop from the inception of your application life; the same applies to corporate IT security reviewers.

In order to keep the "developer-admin" link alive it seems wise to dedicate one or more members of your team to be responsible for interaction with the operational folks. The assignee would probably have above-average shell-scripting skills and administrative knowledge of the operating system(s) and database(s) your application is bound to. When required, this developer would be able to solve or identify problem on his own, instead of calling your SA (DBA)-on-duty. This guy would also be able to provide "administrative" help to his peers with logs, backups, governing user rights and so on.

As with documentation writing, systems administration is something that not all the Developers get excited about, therefore finding and keeping a good Administrator in the team will make your life as a Project Manager much easier.

Programmer - Architect role

The last, but not the least is the Architect role. Someone has to make sure that the application is being developed in according to the corporate and industry coding and architectural standards. Someone has to make sure that the application's design meets all the requirements and provides for future changes, where applicable. There are always issues of compatibility and performance that have to be foreseen and catered for. Finally, someone has to make sure that the application, once deployed, plays nicely with the other apps in the infrastructure.

Surely, there always is a chief architect that could be responsible for all these tasks, but in reality this job is often left to the collective conscience of the development team.

One could also argue that there are design and architecture reviews that are held with explicit goal to catch any of the above mentioned problems. True, even though the reviews are often a simple formality. But why waste time? Why not do it right from the beginning? The idea behind having an architect/designer in the team is that his presence facilitates pro-active process of conforming to standards and making sure that the application would fit. Playing this role implies keeping the Chief Architect up-to-date on what's going on with your application, getting his feedback and staying alert in case there are new corporate IT processes that affect your application. The architect could also run code review sessions and interact with R&D.

Team Leader

One more role I did not cover yet is the role of a Team Leader. In the model, the Team Leader reports to the PM on everything that is in the team's activity scope and is responsible for:

- Keeping project schedule for the team
- Coordinating efforts
- Identifying potential risks and issues and alerting the PM
- Updating the PM on the status of the project and keeping the team up-to-date on what the overall status of the project is
- And, yes, write code!

Laying these tasks upon a Team Leader's shoulders will allow the PM to concentrate on a high-level problems and strategic decisions. There is no real need for the Team Leader to tell his partners what and how to do things, assuming the self-motivated Power Developers are capable of operating on their own.

Role of today's Developers

What needs to be understood here is that many business application developers today are already playing these or very similar roles, but this fact must be recognized and reckoned with. I am merely stressing the fact that an IT organization should be keenly aware of the fact that power developers are today's reality and their "additional" skills must be developed and valued. If you know what tools you've got, you have a better chance to get the job done right - fast.

If all you can do is code – you're history

There are a few more reasons why business application developers should expand their horizons – fierce competition, off-shoring and a tough job market prevails in today's IT market. For today's business application developers that need to design, implement and deploy their products in-house. Possessing great coding skills is no longer enough, nor as essential as it used to be.

Unlike system programmers, application developers are less involved in pure coding. They don't have to come up with super efficient algorithms and write drivers and parsers from the scratch. Instead, they have to be able to connect and integrate off-the-shell tools, libraries and software packages into working applications.

In the world of business application development being just a programmer does not do any longer. Further to being able to code, one has at least to know what tools are good for the job, learn compatibility matrix and keep track of what's going on in the industry. The bottom line and message for application developers here is: if all you can do is code – you're history.

"Alloy people" vs. "Power developers"

In one of the companies I worked for, one of the top managers had come up with a notion of "alloy people". Being one meant that you basically had to be a "Jack of all trades". You had to know what every one else knew and be able to do what everyone did. This was promoted as a skill-lifting program. It seemed more like the management was simply making sure that they could replace you in case you'd decide to leave the company or move you from project to project, in case resource management was not well planned right from the start.

In our age of total outsourcing, the big-ego CEOs are singing the same song – force an identical skill-set upon everyone and your knowledge transfer (and the following layoffs) process will be a breeze. It is quite like building machines and making replaceable standard parts out of the personnel.

My personal opinion is that this approach completely ignores the human nature. It is therefore bound to fail. It actually harms the performance of development teams. In order to keep the skill-set and knowledge level evenly spread, you'd have to keep everyone busy learning tools and apps that they may not ever use. Eventually, the cost of retraining and learning curve time investments will drive your productivity down to the ground. Having said this, I'd like to point out that having coding standards, architectural guidelines and common tool set within organization makes perfect sense.

Development Squads

Now that we introduced the notion of "Powerful Developer", let's see what how to put it to use. The solution, as will be shown soon, comes in a form of an analog to the military unit organization. The military has given us PERT and many other useful management techniques and models, so it seems to be reasonable to turn to it again.

In the military the individual soldier is the basic building block of all Army. The Power Developer becomes the basic acting unit of an IT department. In the Army, the next smallest unit is called a squad – typically a group of nine to ten people led by a Sergeant. The size of a squad is dependent on its function. A squad would normally have:

- A guy with the radio
- A gunner team (two guys)
- A grenade launcher
- Four to five riflemen.

Going by analogy, it deems to be interesting to suggest that something structurally similar to the military squad should be formed in IT, and here is where power-developers come to be used.

So let's group together

- A team leader (becomes a squad leader)
- A programmer-business analyst
- A programmer-architect
- A programmer-document writer
- A programmer-administrator
- Throw in four to five coders (yes, we have said that this kind's dying out, but they're still around!)

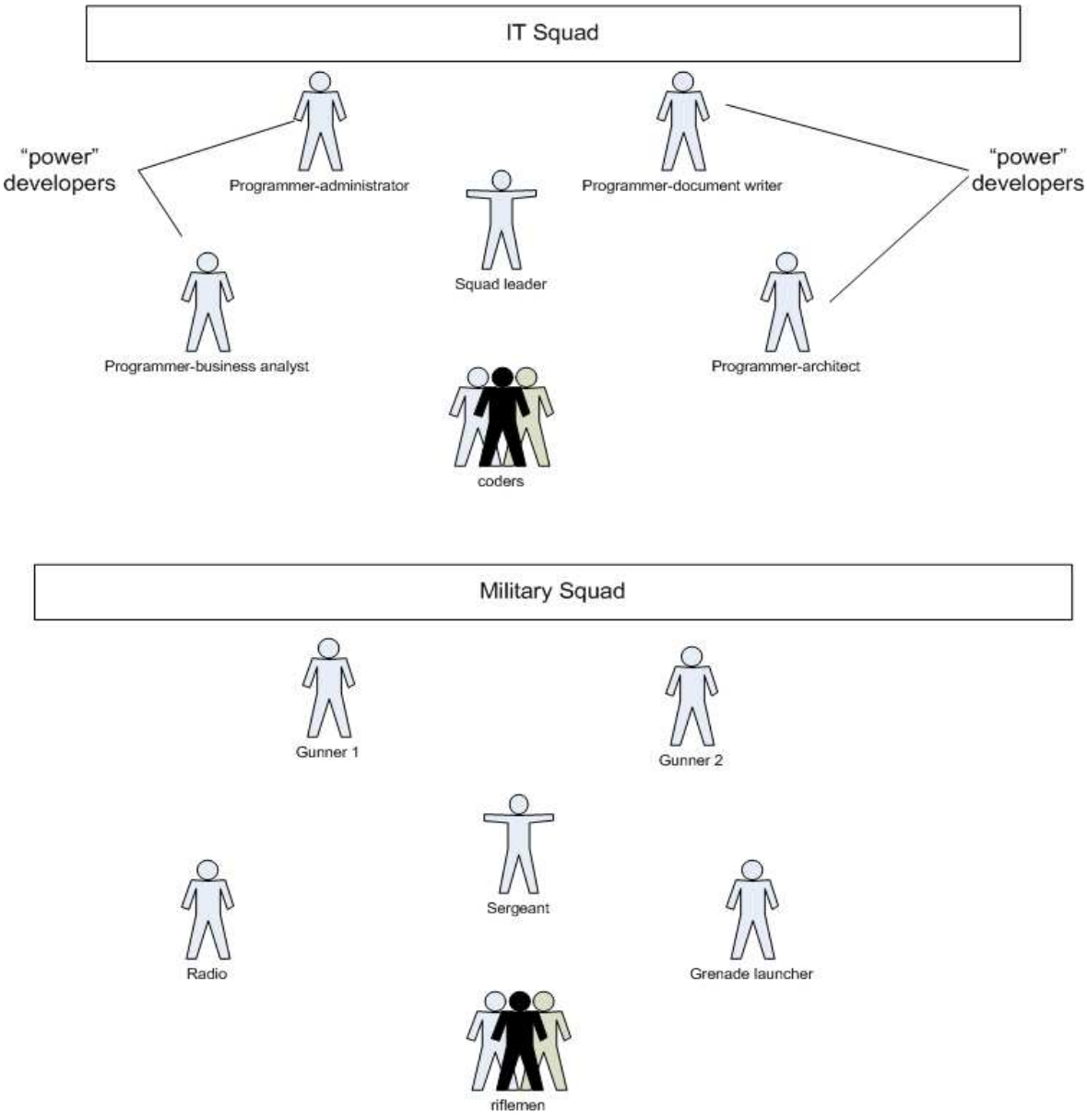
Look, you've just got yourself an IT squad!

Squad Size

The size and staff of your squad will, no doubt, be based on what are the tasks at hand, but it should be kept down to the minimum core – four to six people. If your application is going to be deployed in a very complex infrastructure, you might want to add more Administrators to your squad; if the business and functional requirements are mounting – you would call for more Business Analysts; for applications with technically challenging functionality more Architects will be required – you get the idea.

In real life you do not always have the resources available, but you should try to make the best of what you have. The real challenge is breaking your project down to tasks in such way that the number of tasks matches the number of squads. Alternatively, you could have one squad working on the entire project's tasks sequentially, or you could de-serialize the process by deploying more squads with appropriate skill accents and hand project's tasks down to multiple squads.

Figure 1



Squad Limitations

There are situations when deploying a squad is not applicable, for example – when small bugs need to be fixed. Also keep in mind that the "Power Developer Squad" model applies to development projects, which means that whenever you're dealing with maintenance, you might be better off handing your tasks as a "background" tasks to individual developers.

A few things are worth mentioning here:

- Try keeping your squads together, unless you really must break them. Reassembling squads for each new project drives the team work spirit down and negatively affects performance. It is known that people who have worked together for years make better partners and have fewer communication gaps amongst them. Still, some managers tend to think that resource rotation is a good

thing. My recommendation here - spread the knowledge, don't spread the squads! The ideal situation seems to be the one where there are a fixed number of squads in the department.

- Avoid withdrawing developers from a squad in the middle of the project – this will only frustrate the person and spread feelings of uncertainty around.
- Make sure that the same processes, standards and tool sets are known and used by each of your squads. This will provide you with control over compatibility and let you add members to your squads with no, or little transition overhead.

Assembly point

Now that we have squads covered and we know what the responsibilities assigned to each of the squad's Power Developers are, let's see how this model fits into a hypothetical IT organization:

Figure 2

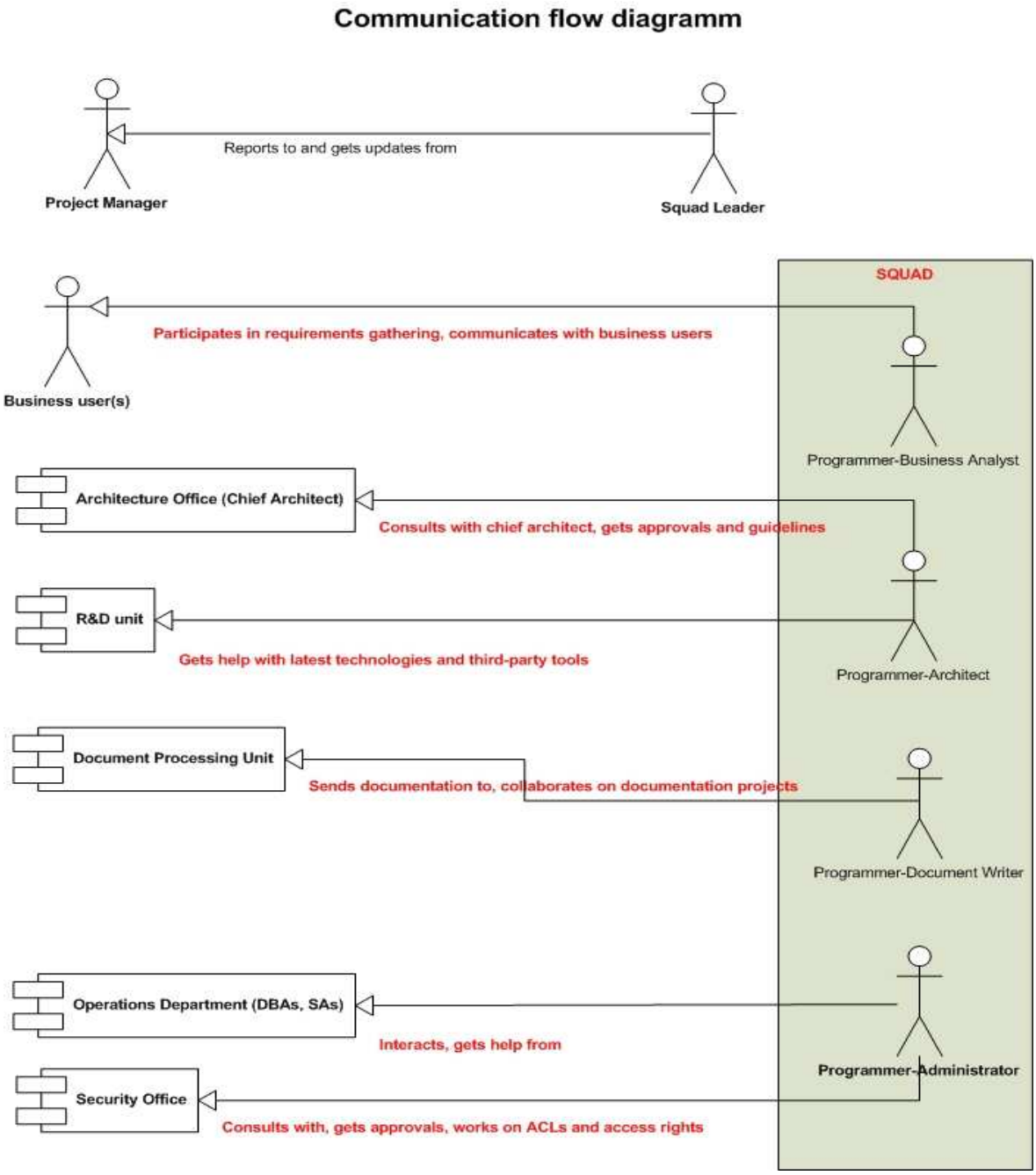


Figure 2 is a graphical depiction of the communication channels that have been described above, but as they say – one picture is worth a thousand words. This is how a single squads' communication with the rest of the world looks like

As far as the organizational structure is concerned, the model does not brings anything radically new to the traditional way of handling reporting hierarchy – squad leaders report to Project Managers (just as traditional team leaders do), Project Managers report to their bosses (directors or executives) and so on.

The important thing is that the Power Developer Squad model should get recognized and understood by the entire organization, so that the communication channels are

open and effective. Let me give a few examples of how the squads go beyond the project scope and enter the organization arena:

- When there is an organization-wide call for documentation gathering, then each squad sends ahead its Programmer-Document Writer guy.
- When there is a milestone architectural meeting, the squads delegate their Programmers-Architects to attend.
- Changes to the company's method of tracking hours and keeping schedules are delivered to the attention of Squad Leaders.
- If the organization migrates to new platform (say, from UNIX to Linux), the Programmers-Administrators learn intricacies of deployment and shell scripting for the new OS.

Conclusion

A good way to see whether the model can be deployed in your environment is to look at your developers from the new point of view. Once you identify who's ready to wear the "Power Developer" hat, see whether you can glue the squads together and establish the communication channels. To achieve success, both Managers and Developers need to start thinking in new terms – the ones of the model's concepts – and act accordingly.

The power developer squad model is what it is – just a model. As I mentioned before, real life will surely make corrections to the model, no matter how hard you try to follow its letter. Even as jobs keep moving out of the country and currently employed just-programmers are getting on the endangered species list, there will still be some pure coders that you'd have to pad your squads with. There will be resource constraints, which will affect project-to-project consistency, size and shape of your squads. You may run into certain corporate rules and strict policies that prevent you from building the model – the potential reasons are plenty. But I think that the power developer squad model is worth considering and it will find its place in IT project management practices.

Denis Urusov is an Associate Director of IT Strategic Computing Organization of Bear, Stearns and Co. Inc, which is a leading global investment banking, securities trading and brokerage firm. Denis has over 10 years of software development and management experience. Denis holds an BA degree in Computer Science from New York University. He could be reached at durusov@yahoo.com

Project Perfect sell "Project Administrator" software, which is a tool to assist organisations better manage project risks, issues, budgets, scope, documentation planning and scheduling. They also created a technique for gathering requirements called "Method H"™, and sell software to support the technique. For more information on Project tools or Project Management visit www.projectperfect.com.au