Team Roles

What's in a Role?

A team needs to divide up the work. It may be tempting to just say "we'll share all the tasks and responsibilities", but that doesn't really work. One reason is that there are too many details to attend; nobody can be thinking about everything, all the time. Another is that diffuse responsibility doesn't motivate action as individual responsibility does (or to put it more colloquially, when everyone is responsible, no one is). So, it is best if each important matter requiring attention be the responsibility of a single, identifiable individual.

On the other hand, there are dangers in compartmentalizing responsibility too far. You may lose a team member before the project is complete, and someone needs to be ready to take over. Even if the team remains the same, the initial division of work is based on reasonable guesses, and some re-balancing will be necessary as it becomes apparent that some are bigger and others smaller than anticipated. Over the long term, an organization will be stronger if its members have been learning from each other, and not only doing what they already do well. These considerations suggest that, while we may want to place primary responsibility for each major task on one person, it is a good idea for that person to share knowledge of their work with someone else who can, if needed, take over part or all of their work.

My suggestion, then, is that two people be assigned to each major project role: One primary, and one secondary. Responsibility rests mainly with the primary occupier of a role, but the primary and secondary together are responsible for making sure that the secondary is ready to take over if needed.

Suggested roles

Here are some typical roles in a software project. Some may be split or combined, and others might be added. The important thing is just to make sure that, one way or another, all the major tasks are covered in some way.

Manager

The project manager is chiefly responsible for the project schedule (and, in an industrial setting, the budget). The manager does not need to be the "boss", though in industry it often works that way. The key thing the manager needs to be paying attention to, throughout the project, is the project plan and how actual progress compares to the plan.

System architect

The system architect is responsible for understanding the overall structure of the software system: How it is divided into parts, and how the parts fit together. A good, modular system design allows most developers to think just about their own part, most of the time. The architect is constantly keeping an eye on the bigger picture of how things fit together.

If the project manager and system architect are different people, then they will need to communicate regularly. In a sense, the manager is always looking at how pieces of the development process fit together, and the system architect is always looking at how pieces of the software system fit together, and any disruption in one is certain to affect the other.

Requirements analyst

The requirements analyst is responsible for understanding and communicating the system requirements. The system requirements define, as precisely as possible, the functional behavior and qualities (e.g., performance, ease of use, etc.) that a system must have. The requirements analyst is responsible for identifying the system stakeholders, eliciting requirements, establishing priorities among requirements, and documenting them in a way that is access able to the key stakeholders. Note that this does not mean that the analyst makes the decisions about what should be requirements and their relative priorities, rather he/she determines these in collaboration with the stakeholders including other team members.

Quality control

Someone needs to be making sure, all through the project, that quality goals are likely to be met. This might include producing a good integration and test plan and making sure it is executed, managing design and code inspections during development, requiring good unit test suites with each module, etc.

The role of quality control is a good illustration of the difference between taking responsibility for some important aspect of system development, and doing all the work associated with that aspect. For example, suppose the quality plan includes insisting that each newly developed or modified module be accompanied by a good unit test suite. That doesn't mean the person primarily responsible for quality control writes all the unit test cases or even inspects them. It is more likely to mean that they establish (in consultation with the team) standard practices and expectations for producing and reviewing unit test suites.

Technical documentation

Technical documentation is documentation written for developers and maintainers of a system, rather than documentation written for users of the system. This may include everything from formal requirements and design documents to README files and comments in program code. As with quality control, the person who takes primary responsibility for technical documentation is not likely to do all the work him- or herself. The responsibility is for making sure it gets done well and on time, probably by coordinating the work of several people.

User documentation

User documentation is documentation for users of a software system. Often it involves more than one group of user. For example, if one were producing a web-based application, both end-users of the system (using the service through a web browser) and administrators of the system (configuring and running the service on a web server) would be users, and they would require quite different documentation. User documentation may include conventional user manuals, online help, web-based documents and FAQs, and perhaps other forms of information.

User interface

Ideally, a software project would have access to usability experts in a usability lab. Often we don't, but it's still worthwhile to have someone take special responsibility for consideration of interface issues. Decisions in other areas often have usability and user interface consequences that are easily overlooked; the UI master is

responsible for making sure they aren't overlooked.

Programmer

The programmer is, as the saying goes "where the rubber meets the road." The programmer is responsible for turning the need for system capabilities into something a computer with actually run. It should be clear, however, from the other roles, that the programmer cannot act independently. Rather he/she is responsible for realizing, as closely as possible, the requirements and designs developed by the team. This helps ensure that what the programmer produces is what the stakeholders actually want.

Configuration control (build-master)

When several people are building and modifying parts of a software system, it helps to have someone in charge of keeping the changes coordinated and making sure the build process continues to work. As with other roles, part of the build-master's job is to establish practices for others, and part is to administer the version and configuration system. In large, complex projects, version and configuration control can be a huge issue, requiring a "configuration board" made up of individuals working on different parts of the system. For a CIS 422/522 project, it is much more manageable, but you will find that coordinating changes still becomes a challenge as deadlines approach.

Mapping Roles to People

The roles described above (and others you may identify) needn't map 1-1 with people. Some require more time than others. Some may be combined, and they may be divided up in different ways.

One combination of roles that seems to work well is user documentation with user interface specialist (for projects that don't have access to a dedicated user interface laboratory and usability experts). This is is because there is a natural synergy between explaining a system and making a system simple and intuitive to use. It's very hard to write a good user manual for a system that lacks a coherent conceptual model. A practice that I have seen work (but which, sadly, is not often used) is to draft a user manual before producing the software that it describes — that is, treat a user manual as a kind of design specification document for the user-visible behavior of the system under construction. It will change as the project develops, but so will any other form of design.

Note that I have not listed "programmer" or "developer" as a role above. That is because, in most cases, several members of your team will share programming responsibilities. The principle of making sure a single person is in charge of each task applies not to programming overall, but rather to each individual programming assignment ... and responsibility for making sure that someone is responsible for programming each part belongs to the project manager, in consultation with the system architect.

E-mail: Email instructor
Based on the open source Sinorca design
with content from Prof. Michal Young and Prof. Anthony Hornoff