XXX XX XX Assessment Report

(report date here)

Johanna Rothman

Rothman Consulting Group, Inc.

781-641-4046

jr@jrothman.com

www.jrothman.com

(logo removed)

Section 1: Project Description

Strategic Goals:

Assess how the XX Line plans, manages, and executes the activities from requirements through delivery of the products. The goal of this assessment is to understand what the next steps could be for improving productivity and where the line should put its efforts. I had two questions driving my interviews and metrics analysis:

- Is the necessary work performed?
- Is the necessary work performed at an appropriate time?

Scope:

I looked at these areas:

- Project management
- Requirements elicitation and definition
- Development practices
- Testing practices
- Tools
- Management practices
- Metrics

Assessment Participants

(removed)

Section 3: What's Working

For the most part, people like and respect each other.

Development and QC work well together.

People are willing to help each other.

People like working at XXX.

The managers have excellent relationships with each other within (one site).

Some of your managers are highly talented at management.

People are honest with their managers and peers, inside their groups.

You successfully release lots of product (internally and externally).

Some projects implement by feature and show the customer progress.

Some projects do preliminary integration, to see if the interfaces will work.

Some people do continuous integration.

Starting integration before the Integration group seems to be working nicely.

You have some situational code ownership.

Some projects included the project delivery people in their project meetings.

You are starting to use portfolio management for some of the groups.

You do project retrospectives and learn from them.

The line is organized—people know what to work on.

People have relatively small tasks to work on (close to inch pebbles), so they accomplish more.

Section 4: Assessment Findings

There are findings for:

- Software project management
- Requirements
- Development
- QC/Testing
- Metrics
- Management
- Other observations

Caveat: I believe that I either heard these findings from people or read them in documents. However, the assessment could be incomplete due to the compressed time.

Findings for Software Project Management

Not all projects are equal. Some projects are quite small and short. Currently, the project documentation for short projects can take almost as much time as it takes to finish the project. Project documentation needs to base on the level of risk in the project, as well as the time involved. It's possible some of your very short projects are not "real" projects. Here's Max Wideman's project definition:

> "A novel undertaking or systematic process to create a new product or service the delivery of which signals completion. Projects involve risk and are typically constrained by limited resources."

Using this definition, some of the XX Line work can be managed with a brief plan and a checklist. I don't know enough about some of these short projects, but anything that's a one-off from other work and is very short in duration may not need all the current project documentation.

Insufficient project coordination with Project Delivery. I don't know who is supposed to do what because I haven't discussed enough with the Project Delivery folks. But it's clear to me that Project Delivery does not explain the relative ranking priority of all the projects in the department, nor do they know when the customer will use the product. The implication for project management here is that all the project managers have to be proactive about working with Project Delivery, and definitely not assume that Project Delivery will manage the projects. Project Delivery does not manage the projects.

In addition, the system architects who define the architecture don't participate on the projects, so they don't know if their architecture is actually going to work. (Frequently, there are changes to the architecture.) And without the Project Delivery coordination, the requirements arrive late.

No use of release criteria. Release criteria are the vital few criteria necessary for a reasonably successful product, not the list of all requirements. Especially for projects under time schedule pressure, I use release criteria to know when a project is done, and as a tool to discuss with the customer (or Project Delivery) what they really want and when. You might find that some projects do not have the same urgency as you think they do, or that the customer will take things in stages. Without release criteria, you have no technique to manage internal or external expectations.

The technical staff performs little or no task estimation. Managers do many of the estimates (when estimates are done at all). This lack of technical staff practice of estimation means the projects will take longer (because no one owns the estimate), and that it's hard to know when each project should start and end. In reality, you timebox development to the iteration size.

Learning to estimate is a key skill developers and QC staff need to learn before they become group leaders or project managers. I can't tell if anyone here actually estimates the durations of a project, or if the date is just given to you.

If you receive projects with due dates, then it's incumbent on the project managers to negotiate with Marketing or Project Delivery to discuss what they will provide when. Or, XX and his managers can discuss when to start the project and how to staff it so the functionality is provided in a reasonable time.

Project management is optimized for small in-the-same-site groups. Nor does the project management work intra- or inter-department. This does not fit your need to coordinate with (different locations).

Requirements responsibility is unclear on several levels in (a site). It's not clear who decides which defects are fixed when for a particular customer. Nor is it clear who's supposed to write the component spec. All of these people write component specs: Project Delivery project manager, system architect, R&D project manager, and lead developer.

Some iterations are 6 weeks long. If you had integrated/concurrent QC, it wouldn't matter if the iterations were 6 weeks long, but 6 weeks makes it that much harder to isolate problems.

QC occurs after iteration complete. Normally, in a lifecycle like the RUP (which is where your phase description comes from), QC would start as soon as some code is checked in. Instead, you're using more of a spiral lifecycle. This occurs partly because the developers don't know/can't depend on which QC person they will have for the project, and partly because the QC people are assigned to so many projects they would have to multi-task even more to do QC during an iteration.

Your iterations are really small waterfalls. Because you don't integrate QC testing into the iterations, the iterations are small waterfalls. A smaller waterfall is better than a longer waterfall, but you don't gain as much of the benefit of iterations when QC is postponed until after the iteration is complete.

Project managers do not see project assessment as part of their job. I can't tell who is the audience for the project web sites. Right now, I'm not sure who gains the value. Normally, the project web sites would be for the use of the project teams and their managers. However, the project managers do little to maintain the project web sites, and don't gather the data (qualitative or quantitative) that would make the sites useful.

You collect virtually no data to see the project's progress. You collect some defect data, which is insufficient to see the actual project state and to steer the project in another direction. This means you don't have data to support needing more people, more time, fewer features, or allowing more defects, or any other tradeoff you might make. The reason the projects are successful is the intensive work by the technical staff, technical leads, group leaders, and managers.

The Development-to-QC ratio is inadequate for what the developers do. If a project does not have sufficient QC resources assigned, the project manager can choose to manage the project differently. But right now, all the projects are managed in a way that expects a certain level of QC engineers, and those people are not always available. Admittedly, this is a non-trivial problem because it involves negotiating with Project Delivery and possibly Integration.

Sometimes, (A DEPARTMENT) not a partner in testing, but an adversary. (A DEPARTMENT) seems delighted when they find problems, which surprised me. The way to make (A DEPARTMENT) a partner is by suggesting intermediate drops, by keeping them apprised of project progress, possibly even by inviting them to project meetings.

Findings for Metrics

Insufficient use of proactive metrics. Your task lists help you accomplish the projects, and having short iterations means you don't need Gantt charts. But you have no way to see the entire project's status. I prefer using a project dashboard, but even if you don't want to measure "everything" you do need to start measuring some things to see if you're making the progress you expect.

Insufficient use of qualitative metrics. Even if you don't want a project dashboard, you need a testing dashboard, to explain what's been tested, what hasn't, and what's left to do.

Only defect metrics available. You currently use only the standard metrics from corporate SQA. Those metrics are not adequate for really seeing the project state; SQA's metrics are based on looking across an organization, not inside a project. You can only see averages with those metrics, not trends, which is what's important to a project manager. As a project manager, I need to know this information:

- When the project really started.
- When the project was fully staffed and if the staffing increased/decreased. I change my estimates if staffing increases/decreases.
- Velocity charts for seeing how many requirements the project team is able to do in a time period. Correlate this with when people are pulled off/added to the project and you would be able to see the productivity effects of multi-tasking.
- Defect trend charts: how many defects are found week-by-week, how many are closed, and how many remain open. If QC is not finding defects (because they are not assigned to the project), you will see a hockey stick graph of new defects. If you see a hockey stick for each iteration, you know you have to change development practices.

Findings for Management

The management team of the line does not have a unified view of all the work in the line at every level (insufficient project portfolio management). Some of the group leaders have the details for their groups. But there is not a Development (or even each Development group) or QC or Line perspective of all of the work across time.

Especially if you're "never going to say no to work" you need a line view of all the work. Without that, you are at risk for—and have—assigned people to too many simultaneous projects. You have options rather than so many insufficiently staffed projects: staggering the start of the projects (you never say No, but you say When), pushback on dates to Marketing, using release criteria to force the decision of what's most important, staging the delivery of the product (saying How Much), or not assigning QC (saying How Good).

Without this unified view of what projects should be staffed when, it's too easy for project managers or development managers to cross their functions and assign QC people (or development people) tasks that their managers don't know about.

People are assigned to too many projects at one time. People make significantly less progress on any one project when they multi-task on several projects. I don't know how to estimate how much time you lose to context switching, but I know the cost is relatively high. And too often, people don't have one #1 project; they have several #1 projects.

Some of the costs of context switching are: actual time it takes people to move to another task, technical debt in the projects (because the first thing to go is code review and unit test development), and a longer final QC stage, and a longer Integration stage.

Without a clear management decision and without management guidance, the technical staff will do what they think is right. Their decisions may not agree with yours.

QC and development engineers are not fungible. Right now, the managers juggle people among projects, so if a person started one project, and was pulled off, that person might not return to finish the first project. This is most obvious in QC, but also happens in development. Each person has a unique set of knowledge and skills. Some people are more versatile than others. When you pull people off projects and reassign them somewhere else without allowing them "immerse" time, they don't learn the system; they just do the bare minimum.

You have choices about multi-tasking and multi-re-assignment. One choice is to rank all the projects. One choice is to specifically delay the start of some projects. Another choice is to not assign QC people to some projects. You don't have to staff all projects with QC at the expense of all projects; instead you can choose which ones to staff and which ones not to staff.

Dev: QC Ratio means you have to make choices about how you develop. You may have to keep this ratio for a while. In that case, you need to staff fewer simultaneous projects, decide which projects to *not* QC, or you could change development practices. For example, you could choose to do pair programming with test-driven development on some projects and not need a QC person, depending on your relationship with the customer. You might choose to staff fewer projects with more people to finish them quickly. But this is a management decision to make.

Not every group leader holds regular informal one-on-ones with his or her staff. When they do hold one-on-ones, they focus on short-term deliverables, not feedback, mentoring, or long-term employee growth and/or career planning. Having a discussion once or twice a year about people development is not sufficient for people to grow quickly.

Roles and responsibilities are not easily matched with titles and what people want to do. The issues of whether to be a staff engineer or a group leader or a program engineer or a product manager or a project manager or a manager are complicated. I can't tell what title correlates with which role(s). It appears that one of the causes is that people keep receiving promotions if they've been here long enough. That's a problem because it causes title inflation, and can only be handled by HR.

The problem for the line is how to work in a matrix organization. If the project managers (who happen to be development mangers or group leaders or senior developers) can assign tasks to the QC people (and I think they should), how does the QC Manager know what that person's assignment is? It's not clear how the QC Managers deal with the requests for QC people on which projects if they don't know their current assignments.

Learning the technology is difficult. Many people only know their small piece of the product and are not aware of how to learn about the rest of the product or the rest of the products in the line. This occurs in XX and in (a site). The managers don't feel as if they have the time to spend help people learn the technology across the line. The result is that people are pigeonholed and continue working on products for years.

This line has a large number of "staff" people. I counted 13 staff people for this line (architects and SQA staff). I can't tell if this is part of the title/promotion problem. But not everyone is really a staff person.

Some of the architects are doing real projects (I think), and if they are, they should be with the project groups, to make sure they transfer their project knowledge in bits and pieces. Having that large a number of people unavailable for the general pool of developers and projects makes it difficult to fully staff the projects. And having the architects doing projects without transferring them into the projects means that Anatoly has too few architects on his intensity high urgency projects.

Some of the architects don't sit with the projects. If I understood correctly, XX's group does a combination of early R&D projects and development projects (more than prototypes). I would move those people doing development projects into the project teams. Some of the architects work with the development teams, but not consistently or directly with the development teams. Some of the systems architects do not work directly with the project teams. In my experience, the most successful projects have architects who fully participate in the project team.

Managers need to choose how to spend their time. Group leaders and managers have many competing tugs on their time: project work, email, vmail, meetings (at least). I heard many comments about the management meeting last week.

There are generally three parts to a meeting: information dissemination, discussion, and decision-making. Whether it's a project meeting or a management review meeting, I would separate the three parts, and decide whom I wanted in each part. It's possible I wouldn't change my mind about how many meetings or who was invited, but I'm sure that not all the people who attended

Section 6: Gathered and Derived Metrics

I thank XX for his help in gathering metrics. Unfortunately, I was not able to gather enough data to derive the metrics.

This is the data I was attempting to gather for each project:

Total LOC (product)

Total LOC (test)

Total person-months

Calendar month duration

Total found defects

Total fixed defects

Total remaining open

Total rejected fixes

FFR (# bad fixes/total fixes

Defect density (/1000 LOC)

Defect density post-project

Defect escape rate

The reason I look at LOC for both development and test is to see how "big" the product is. LOC is not a totally accurate size picture, but it gives me some idea. As far as I can tell, none of your projects are software-in-the-large, which is generally 5Million LOC or more.

I use total person-months to see how complexity changes the development effort. And I use calendar months to see how much multi-tasking occurred (relative to other projects). The more multi-tasking, the more defects the project team should suspect.

Reviewing total defects found helps you know whether your defect-finding activities are working (during the project). The total remaining open is an indication of technical debt. I calculate Fault Feedback Ration (total # bad fixes/total # fixes) to see if the developers are in trouble during the project and to see how much technical debt there could be at the end of the project. Defect density will tell you how effective your developer testing is. Defect escape rate will tell you about the experience your customers have in using the project.

The most telling part of the metrics gathering process was how difficult the metrics were to gather. They are not all in one place per project. There is no query (currently) for ClearCase to provide the defect data I wanted. The project managers would have to go back through their task lists and Outlook to see when the project really started and ended. I'm not sure the time-tracking system can give the person-month data, especially not with all the multi-tasking.

Section 7: Specific recommendations

This section has recommendations for:

- Software Project Management
- Requirements
- Development
- QC/Testing
- Tools
- Metrics
- Management

Recommendations for Software Project Management

Short Term Recommendations

1. Assign people to just one project at a time. If you can do this for a full iteration, that's great. Even if you can only maintain the project assignment for a week, that's better than asking people to work on more than one project at the same time.
2. Decide if all your projects require the same documentation and create shorter/smaller docs to deal with the different projects. See if the adaptations people have made to the SDP would better serve more projects.
3. Start discussing release criteria on new projects or just-started projects with Project Delivery to know the relative priority of the feature set/defects/schedule and how to make tradeoffs.
4. Keep all your iterations to no more than four weeks.
5. Start with a pilot project and keep the QC person on the project the whole way through, allowing that person to start testing during an iteration.
6. On one project, try implementing by feature (as small as possible) and having the QC person test that feature, to see if you can move to more of an iterative development cycle rather than small waterfalls.

Long Term Recommendations

1. At one of XX's management meetings, discuss what data would make it easier for you to manage projects. Decide how you'll collect that data and where to post it.
2. Choose a pilot project and estimate it. Update your estimates all the way through and see how close you are. Decide what to do about estimating projects in the future.
3. Integrate your projects as much as possible with Project Delivery, so they can see your progress.
4. Once you have people working on only one project at a time, start updating your risks and managing them.
5. For projects without QC staff, decide what practices the developers need to use to succeed. Release criteria might help here.

Recommendations for Metrics

Shorter-term recommendations

1. Measure your cost to fix a defect throughout the project, so you can decide about how much design review, code review, and unit testing you want the projects to do. The higher your cost to fix a defect, the more developer testing you want to do.
2. I would measure the support load on the different developers in the different groups and see if you want to change your approach to support.
3. Define the data you want to collect. Decide what you can collect now and what you'll postpone for a while. I recommend you gather this data:
    a. Schedule estimates and actuals
    b. Estimation quality factor
    c. When people (with the appropriate capabilities) are assigned to the project vs. when they are needed
    d. Requirements changes throughout the project
    e. Fault Feedback Ratio throughout the project
    f. Defect find/close/remaining open rates throughout the project

    I would start with at least this much:

    Start date, projected end date, number of times end date changed, # person-months, # calendar months, velocity in terms of requirements completed over time, and defect find/close/remaining open during the project.

    See the additional project dashboard article.

Longer-term recommendations

None for now. First you need to see if the data you do collect is reasonable and is helping you accomplish what you need to accomplish.