

2008

PACIFIC NW SOFTWARE
QUALITY
CONFERENCE



COLLABORATIVE
QUALITY

OCTOBER 13-15, 2008

CONFERENCE PAPER EXCERPT
FROM THE

CONFERENCE
PROCEEDINGS

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.

Collaborative Quality: One company's recipe for software success

Alan Ark
QA Manager
Compli
arkie@compli.com
<http://www.linkedin.com/in/arkie>

Author Bio:

Alan Ark is currently the QA Manager at Compli, in Portland, Oregon. Alan has gained tremendous experience working for Unircu (now Kronos – Talent Management Division), Switchboard.com, and Thomson Financial – First Call. While at Thomson, Mr. Ark presented “Euro: An Automated Solution to Currency Conversion” at Quality Week '99 (<http://www.soft.com/QualWeek/QW99/>) detailing how an automated test suite written in perl was used to save the company great embarrassment. Currently, Alan is teaching his staff how using Ruby can help speed up the testing process. Alan holds both a BS and MS in Electrical Engineering from Northeastern University in Boston, Massachusetts. Interests outside of work include playing golf and listening to Jimmy Buffett songs.

Abstract:

Building software is not an easy process. There are many obstacles that can get in the way of delivering a great product in a timely manner. The intent of this paper is to share one company's experiences with getting better software out the door. This paper does not give you the silver bullet, but hopefully will present some ideas that you can take away for your shop. We don't formally subscribe to any one methodology, but we endeavor to use Agile processes, and are tweaking our process as we learn from each subsequent release that we work on.

While we've had some great results, the focus of this paper is on the little things that gets us those results. The ideas in the paper are presented in more of a cookbook style where the final results are more about the preparation and ingredients, rather than sticking to any pre-determined step by step process. The execution of the process is an important step, but without good ingredients, sometimes you get less than stellar results. We are always adapting to lessons learned with each release, but our base recipe is fully cooked. Think of this presentation as family favorite recipe that you can add your own flair to.

Background:

Compli was founded in 1999 to create the market's first comprehensive web-based compliance management system. In late 2004, a consultant was brought in to evaluate the technology used by the company and assess the product as it was. As a direct result of the analysis, a whole new engineering team was brought in to take Compli to the next level. What that meant was that the current engineering team inherited outdated .ASP code that was not scalable, not maintainable, but according to the previous CTO, contained **ONLY** one bug. Previously, releases to production usually entailed an all-hands-on-deck approach for handling customer calls for the next several days. There were no formal software processes being followed. It was a QA nightmare. There were no acceptance tests defined. There were no test plans in existence. There was no real documentation in existence besides an outdated user manual. This was life in 2006. Let the fun begin.

Base Ingredients:

The base ingredients are just the most basic things used at Compli to build software. It is very likely that you'll need to use these very same ingredients as well.

People

When it comes down to it, a business is only as strong as its people. Each company will have different needs and wants from the people that work within its walls, but without the people, the business is nothing. Teams come in different sizes and shapes, but there are commonalities across different companies. Common job functions, common responsibilities, yet each team is distinct in what it wants to achieve, each company in what product to deliver. People can be found just about anywhere, but it is upon each team to know what kind of person that they are looking for.

Trust

Trust can be defined as being a person on whom one relies. If you have trust between team members, it will be easier to work together. Without trust, it will be nearly impossible to make progress. Unfortunately, trust is not something that can be bought from a store. You must grow this on your own.

How do you earn trust?

Respect

Respect is esteem for or a sense of the worth or excellence of a person, a personal quality or ability. Like trust, one cannot buy respect at the store. It is another item that you must grow on your own. Luckily, if you find trust, then respect is usually lying right next to it.

How do you earn respect?

Communications

Communications is a generic term used to describe how one thing interacts with other things. In the context of communications as an ingredient, it benefits everyone to find the crispest communication possible. Each communication that you make or create is a direct reflection on you.

Focus on what you are trying to convey.

What's the goal that you are trying to accomplish?

A cluttered message usually leads to confusion.

Infrastructure

Infrastructure is one base ingredient that you can buy in a store. These items are more examples of things that are usually common across companies in different fields, yet how infrastructure is implemented will vary widely based on a number of individual factors – each which is unique to your specific company. Some of the major pieces in use at Compli include the following.

- Development Environments
- QA Test Clients
- Project Documentation Repository
- Code repository
- Bug Repository
- Continuous Integration (CI) system
- Automated Testing

Execution:

With the base ingredients in place, these are the steps that we take to start baking our software projects.

Access

Even if you had the best, most expensive, magical tools that solved all of your problems, they would be useless unless you could access them. Some of the items that we have access to at Compli include.

- Engineering tools
- People
- Project Status
 - Development Status
 - QA Status
 - Bug Status
 - External Teams Status
 - Progress against the schedule.

Nurture Communication

Communication must be nurtured to ensure that the right message is being sent. It must be used correctly or the risk for misunderstanding – a miscommunication – will increase. We talked briefly about what makes for strong communication. Some examples of how we facilitate communication between team members are listed below.

- We sit in the same area.
- Single repository for core information about the project – salesforce.com
- Minimize having important communication go out thru only email
- Wiki and Blogs used to supplement
- Let people know ahead of time what is going on.

Story collaboration

At Compli, we take the following general steps to develop stories.

- Discuss general themes to implement.
- Use Portfolio Management techniques to decide what to execute on.
- Create and review stories
- Estimates on the body of work
- Establish acceptance criteria on stories

One topic that might be new for readers of this paper is Portfolio Management. This was introduced to Compli by Mark Lawler. Portfolio management can help articulate what the real priorities are. It takes into account the business needs, perceived risks and product expectations and presents the information in a visual format that brings out the most important things to tackle first. Visual examples of the portfolio management output graphs can be found as Figures 1 and 2. For a copy of an Excel spreadsheet where you can try portfolio management for yourself, please email me at arkie@compl.com. Thanks to Mark Lawler who approved distribution of his spreadsheet.

Risk / Reward

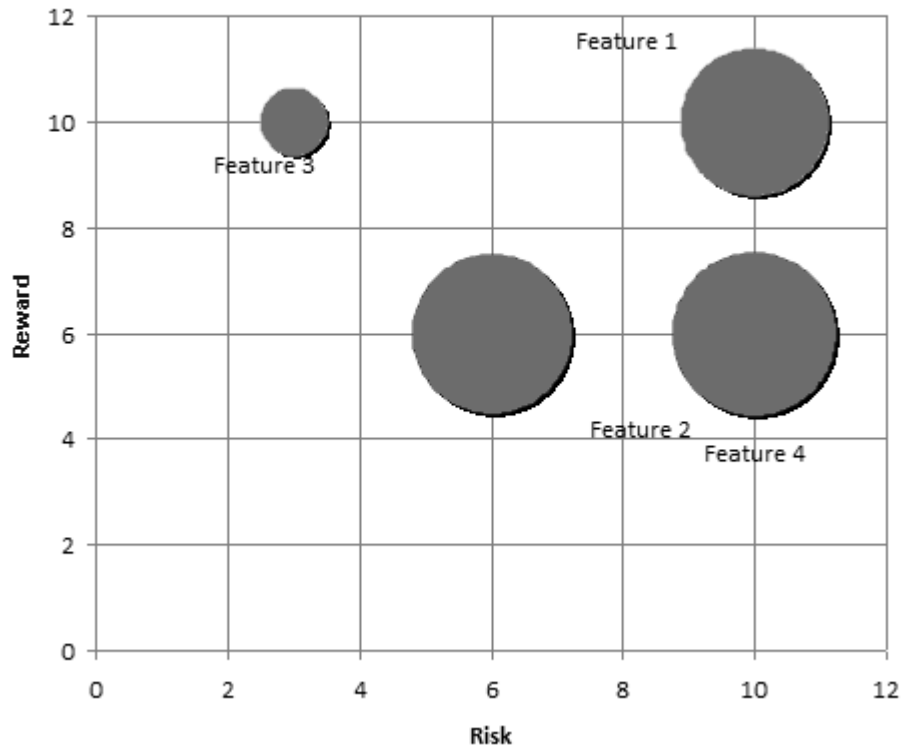


Figure 1 – Risk/Reward graph from the portfolio management strategy

Strategies

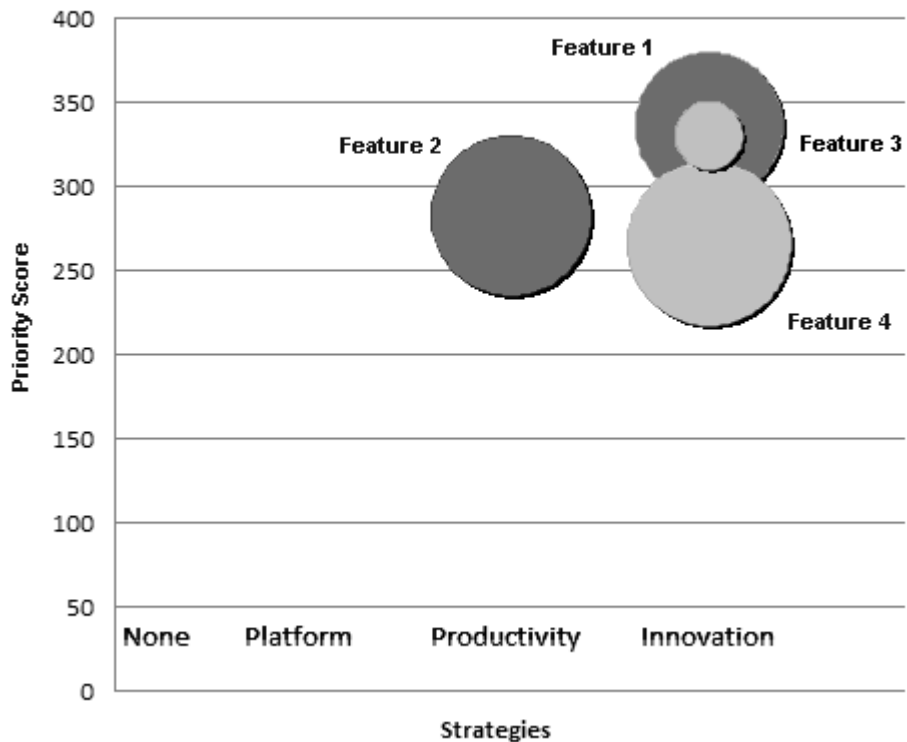


Figure 2 – Strategy/Priority graph from the portfolio management strategy

Metrics

Metrics allow us to measure our progress on the project. What matters is that we tried a bunch of things that we thought might prove useful. The ones that we kept are discussed here. Figure 3 shows a bug trend report that displays all accumulated bugs against a project over time. We have also tried other metrics that we've since discarded because they didn't prove as useful as much as we had hoped.

Metrics we kept include:

- Work estimate times
- Work actual times.
- Bug Statuses
- Bug Trends

Some of the discarded metrics included the following.

- Bugs by Origin
- Bugs per story
- Bugs per feature

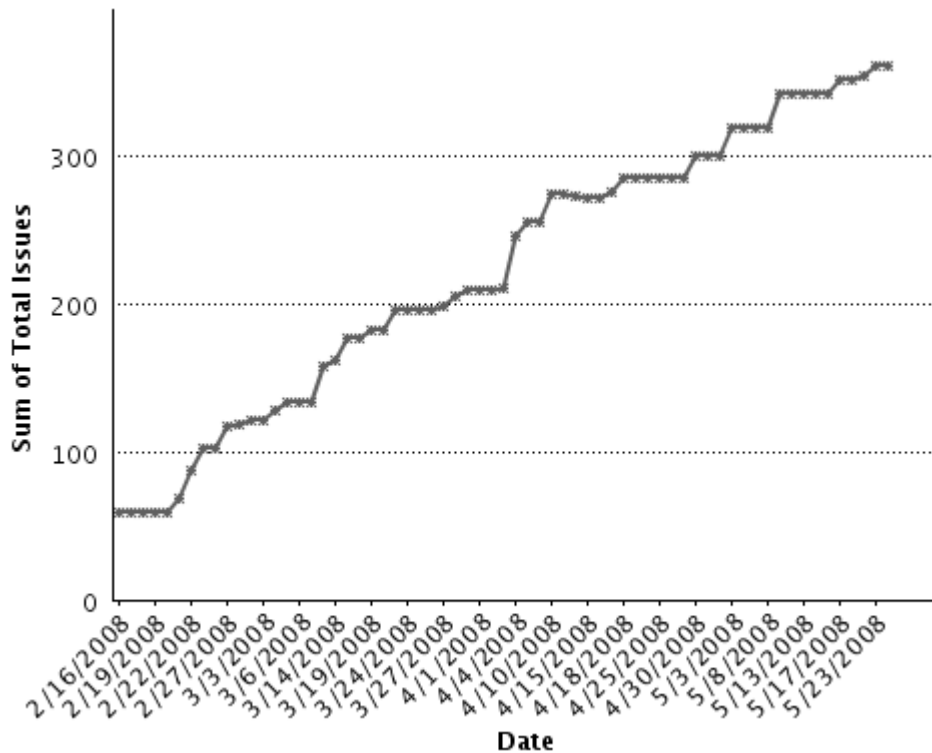


Figure 3 – Bugs accumulated on a project over time

Iterations

Our work slices are called iterations. Each iteration provides a timebox where we try to accomplish specific development/testing goals. Our iterations generally work as described below.

- Continuous Integration of code.
- nUnit tests executed.
- 2-3 week iterations depending on the list of stories to be delivered.
- Daily morning stand-up
- Story demos to the entire product development team
- Coordinated drops into the QA environment at the end of each iteration
- Watir based acceptance tests
- Quick turn around on any bugs blocking the testing of stories.
- Bug Triage as needed.
- Scheduled demos with people outside of engineering to gather feedback on ideas
- Comparison on page load response times.
- Iterations at the tail end of the project also include the following features
 - Usually run 1 week in length
 - Normally bug fixes as opposed to implementation of new features.
 - Practice of what would happen on a release night.
 - Full deployment to QA
 - Full DB migration
 - Web side rollout (web and middleware code)
 - Watir based regression tests
 - Manual regression tests
 - General Availability to fellow employees

Release to production

While the class of hardware in the QA environment is different than what is used on production, each production machine has a QA counterpart. Each rollout into QA can be considered a practice run of a rollout to production. As the project becomes more mature, we execute on full rollouts of database migrations and all build code on the QA hardware. We have a high confidence level that the behavior on the QA systems is very similar as the production system. This makes the actual release to production go smoothly.

- DB migration
- Web side rollout
- Watir based acceptance tests
- Manual verification of “hot-spots” identified in testing of iterations.

Results:

Overall, the results have been outstanding, especially when compared to the experiences of the company pre 2006.

Project duration

- 1 month – 9 months depending on the severity/complexity of the work involved

Major release rollout average 2 hours once ready for QA verification.

Tried this on 10 major releases and 30 hotfix releases in the past 2+ years.

- Only 4 hotfixes in the past 10 months

We have not had a rollback of any release.

The day after a release is a non-event for the company.

- Happy customers
- Happy bosses
- Proud employees

Conclusions:

By using a mix of project management and software development techniques, learning from others at conferences such as PNSQC, and experimentation at home, we have a living process that has helped us deliver high quality software to our customers. We view processes as guidelines to follow, but not rigid rules that must be followed to the letter. Take away bits from others to see what would work within your company. Keep those things that are working, and see what can be improved on those items that are not working so well. Figure out what your specific needs are, and use these ideas to address them.

Don't get boxed into a process

Be flexible. If a change is working, incorporate it in your main process.

If things didn't work out as well as hoped

- How would you mitigate those issues in the future?
- Try things out
- Don't be afraid
- Recognize when things are going well or not so well

If things didn't work, what else could you try to make it work better in the future?

Further reading:

Cooper, Robert G, Scott J. Edgett and Elko J. Kleinschmidt. Portfolio Management for New Products. Cambridge, MA: Perseus Publishing, 2001.

NUnit. <http://www.nunit.org/>

Watir. <http://wtr.rubyforge.org/>

Ruby. <http://www.ruby-lang.org/en/>

CruiseControl. <http://cruisecontrol.sourceforge.net/>

Salesforce. <http://www.salesforce.com/>

Thanks to Launi Mead and Jonathan Morris for their review of this paper.