QA Effort Estimating – CQAA Event 11/13 – Meeting notes

**Team Helene**

**Method of Delivery**

* 90% of the team uses Agile but they have different flavor of agile which is midway between waterfall and agile (lots of the team showed waterfall behavior in inception of projects, automation and performance – QA + Dev )

**Methods of Estimating**

* T-Shirt sizing
	+ Reserved to large projects to pre-estimate a large project
* Points
	+ Points are tied to hours (1 point always represent a day of work = 4-6H)
	+ Points are not tied to hours (relative measure)
	+ Points can be tied to the Fibonacci sequence - 0, 1, 2, 3, 5, 8, 13, 21, 34 which makes it exponential and close to the T-shirt sizing (you put stories in “buckets” of points.
	+ Points are used with agile planning poker
* Hours
	+ In the case where points are translated to hours, the teams use hours indirectly
	+ Some teams prefers hours over points because it allows them to know how long a person will have to work on this – more realistic

**Teams involved in estimating**

* Depending of the maturity level of agile (scrum only for Dev + QA or Dev, QA, Automation or Dev, QA, Automation, Performance, who is involved in the estimation varies.
* Usually stories/tasks get estimated based on who is going to perform the task on the story
* Usually the estimation happens in a group at the beginning of the sprint

**Issues**

* Issues estimating revolve around estimating unknowns
	+ Bugs during the coding of a story (we don’t know what bugs are going to show up after the story is developed.
		- Some companies estimate each bug separately
		- Some companies believe that the estimation of the bug is part of the story and therefore do no estimate the bugs
	+ Performance testing the project (we don’t know what problems the team will encounter in performance testing and how long it will take to resolve)
	+ Teams affecting each other – one team can affect other teams and this is not known ahead of time
	+ Team member availability
		- Once the story is done, the teams work on new stories and therefore do not have availability to go back and fix the story
		- If a team affects a different team, those teams are fully allocated in working on other stories and therefore it is hard to estimate when they will be available and how long their efforts will be
	+ System complexity
		- Depending on how closely the features are tied together, age of the code … they will be more unknown and therefore a harder time estimating how long a feature will take because of potential bugs, complexity of writing the feature and affecting other teams (…)
		- Needs of the customer – 1 small change (few devs hours) can create several hours of QA because it has to be tested on several browsers and a mix of (operating systems, browsers, screen sizes, landscape/portrait options)

**Potential solutions**

* Make sure that all of your organization is agile: Dev, QA Manual, QA Automation, and Performance need to be done and delivered in small chunks – the smaller the easier to manager and to roll back.
* Stories should be decomposed to the smallest item possible for better estimation (more granular)
* Estimation should involve your entire team (dev, QA Manual, QA automation, Performance and we need to let the Product owner know.
* The team seems to prefer hours over points during the sprint (or points that are tied to hours).
* Estimated hours/points should have a buffer calculated over time (estimated vs. actual)
* Definition of “done” should include automated and performance tested and the estimation needs to include those times.
* Overestimate and over deliver

**Team Dai**

**Overall**

* Participants in this group do not use T-shirt sizing or WBS
* Most use points and rarely use hours for stories or epics planned
* Most have 2-3 weeks of per sprint but not every sprint releases to production. With one company releases once a year with 12 sprints per year.
* The allowed points per story run from 8 to 13 from companies in this group. Otherwise, stories will get further broken down.
* Testing estimation happens with all BA, Dev, and Project Manager in one room

**Factors Impact Testing Estimation**

* How development code would impact regression testing
* Effort on integration testing
* Complexity of system/application under testing
* System/application dependency
* Testers’ own technical and business domain knowledge
* Types of testing
* Development’s effort
* Platform supported
* Data preparation
* Testing environment needed
* New technology
* Buffer time
* Issue cycle time
* Clear requirements

**Recommendations**

* Influence development on points estimation
* Everyone on the agile team needs to be in the same room
* Need requirement definition ready
* Prioritize tasks for stories

**Team Prachi**

**Overall**

* This group consists of a mixture of non QA, career QA as individual contributors, and some QA Managers. None of the 12 participant companies used waterfall.
* Teams are using some version of agile, not all teams are fully agile…Very few release code to production every 2 weeks.
* A lot of people felt that teams do not understand agile correctly. That’s why there are various types of agile
* There is also a lot of inertia to implement change in practices which exist over decades.  Advantages of agile were not understood by all companies. One comment was that Agile will not make us faster or cheaper, it will just help built better quality.

**Types of estimation techniques used**

* Story point estimation using the Fibonacci series
* At some places, QA points are not estimated. The dev provides estimates and that determines the sprint commitment. This poses a problem because there could be stories which have little dev effort but great QA effort
* The sequencing of stories appears to be an issue. It was a common experience that QA work in sprint is not staggered but is delivered in the last couple days
* There was confusion on what determines one point. In some teams, 1 point was equal to 1 day, in others it was relative estimation.
* Some teams used separate points for dev and QA, some places used points for the whole team without distinguishing dev and QA

**Recommendations**

* All concerned parties (dev, QA, BA, design) should be part of the estimation process
* Both Dev and QA estimates should be clubbed together and estimate velocity as a team, rather than separate QA and dev velocity
* Scheduling recommendation was that stories with high QA point needs to be tackled first so that QA can complete the committed work within the sprint.
* Points should not have absolute value but relative.
* It is advisable to even include automations points in the QA estimate

**Team Nancy M**

**Overall**

* Only one company in this group was tracking actual time for QA while all others are doing agile and only monitor velocity.
* Estimation was primarily done as a team with story points but for some QA discussed separately and came to the sprint planning prepared with estimates.  Some did it by comparing similar features.  Most teams felt there was flexibility to increase the story points based on higher efforts needed by QA to accomplish items such as compatibility testing and automation work.  This was identified and discussed during estimation.  Some groups are also using the squad approach.
* Many teams take on work outside of just QA.
* Discussed the pros/cons of adding stories into the sprint to cover no sprint activities such as training and other work  or just reducing the capacity of the person thereby reducing the velocity.

**Key items that helped with estimation**

* Involvement in the analysis prior to estimation (story time)
* Early prototypes providing a better feel for what they would be getting
* Up front risk assessment
* Well defined stories and acceptance criteria
* Understanding resource time (days off/work on other projects or non QA work)

**Issues that impact estimation**

* Stories added after the sprint starts – change requests
* Lack of action being taken on retrospective feedback
* Incomplete details in stories
* One company (3 people in this group) kept a spreadsheet to track time for resources to understand estimates but this was done outside of tooling – manual effort
* The rest focus on team velocity.
* Automation was briefly discussed – the teams varied – some included within sprint others a sprint behind – but seems overall the preference was to move the work within the sprint.