SCOPE METRICS

SMARTER KPIS DRIVE SMARTER DECISIONS - PART 1

Scope

Keywords:
Scope
Smart KPIs
Risk Management
Decisions

Abstract: *This paper addresses the need for smarter metrics for **managers to make the right decisions** during a Sprint, a project or a program, but also to make the call **when to ship** a new release in production or to consumers.*

*The vast majority of organizations still make such decisions based on **SWAG** (1) rather than documented metrics that matter. XStudio brings a set of automated and real time **KPIs** (2) to help in this.*

*This paper explains how this works using XStudio's **scope** (weighted requirement coverage, sometimes also named readiness). It presents the basic notions and describes the major advantages that it brings.*

Hence, in this paper, we will discuss only of the requirements coverage. The second paper of this series (part 2) will expand the notion of scope to the specifiability and testability aspects.

Summary

Introduction

Help

Among the many questions that managers need to answer when driving a project, there are 2 recurrent ones that XStudio can be greatly facilitate to answer:

- *How are we progressing so far?*
- *Is this version ready for shipping?*

Many managers rely on little more than burn-down or fever charts, a list of due deliverables and target milestones and the team's "guts feeling".
Quite often a manager's question about "*How is the project progressing?*" is understood as "*Are we delivering fast enough to deliver on time (and if possible on scope and in budget)?*".

This is not quite the same focus.

The second question is about "meeting delays" while the first one is about "scope, quality, maintainability and delay".

Rather than focusing solely on estimating the remaining work, we should be measuring and addressing the following questions:

- At that stage of the project, how much of the **scope** are we covering?
- Are our requirements covered by sufficient **documentation/specification** to remain maintainable?
- How much of that scope covered is **testable** or **tested**?
- What **quality** level did we reach?
- How many critical and important **defects** remain to be fixed?

Answering these (not so simple) questions is possible and automated with XStudio.

At any time, you know the coverage of your Application/Solution/Release/Sprint in terms of scope, tests, quality and documentation.

Whatever your delivery cycle and test strategy is, you can have the same level of information. You can answer the above questions with documented metrics. You can make decisions (e.g. adding resources, adapting scope, taking a less risky test approach...) during the project.

You can also take the right decision when it's time to release: it is no more only about "did we address everything that was requested?", but "are we delivery the right scope with the sufficient quality and will we be able to support and maintain this?"

Scope

In the following, we'll take for example an agile project - although this applies in the same way to unified process project, V-style and so on.

Each story goes through 3 steps:

- **New:** you have identified the story title, you know who can help refine the need
- **Ack:** you defined how requesters and/or users will validate that it answers their needs
- **Approved:** everybody seems to understand the same thing and you have captured detailed information (e.g. computation rules, GUI layering, input and output data...)

Figure 1: Workflow

At the start of each Sprint, it is good practice to have 100% of the Sprint backlog identified or defined and possibly detailed to some degree. At least some **conversations** (3) should have taken place between all stakeholders during the Sprint planning.

When using XStudio, you manage Stories as XStudio "requirements". For each Story, there is one requirement.

In such agile context, one of the numerous practices is to create a SUT (system Under Test) per Sprint (e.g. "my solution v1.0s5" for V1.0 sprint 5). You then link the requirements (Stories in our example) to the SUT. This defines **the scope of the sprint**.

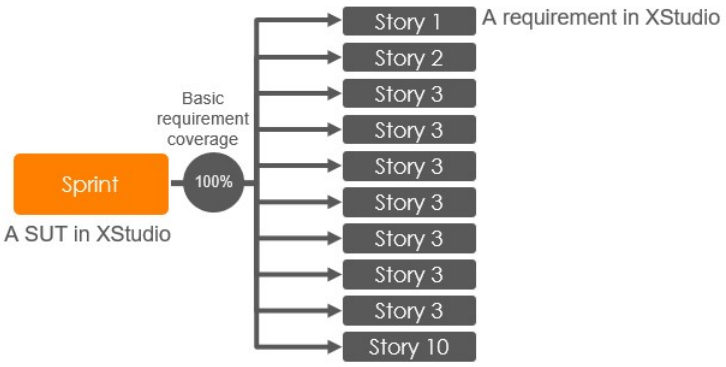

Figure 2: Basic requirement coverage

You then get the **basic requirement coverage** as an indicator. This is the number of requirements (in this example "Stories") that are to be implemented into the SUT (here "during the Sprint"). In our example, we have linked 10 requirements to this SUT and we have a **basic requirement coverage** of 100%.

Because this does not always reflect reality, XStudio allows to manually "correct"" this coverage level. For example, you could state that you are in fact plan expecting to get 20 Stories in the Sprint, hence the current requirement coverage is 50% - said otherwise you are still missing 10 Stories. This is already better than the basic requirement coverage result. We call it the **corrected requirement coverage**.

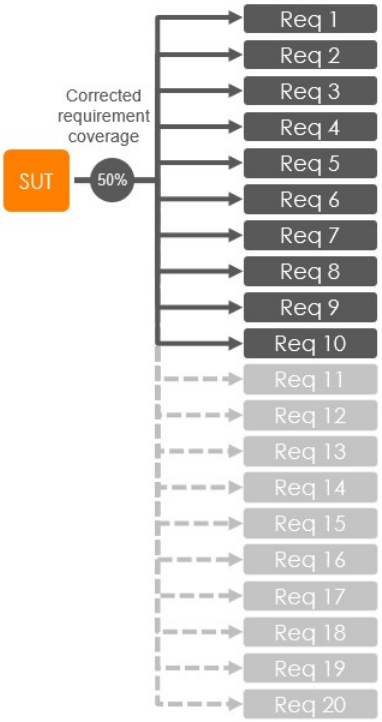

Figure 3: Corrected requirement coverage

But it still wrongly answers our question: "*How are we covering our scope so far?*" Because in such case, we don't know if any of these stories can in fact be designed and developed. Still, this is a useful but insufficiently precise metric. **The corrected requirement coverage doesn't support well our decision-making process.**

By contrast, XStudio's **weighted requirement coverage** is automatically determined **based on the requirement status and priority of each requirement.** As development team progresses the requirements throughout conversations and confirmations workshops with the other stakeholders, the **weighted requirement coverage will also automatically progress in real time.**

You can know confidently answer the question: "*at 50% of the elapsed time of the sprint, we have 70% of our scope that is defined - we are in the green zone on that aspect*" or "*2 days before the sprint review we have covered 80% of the scope! We have to push back 2 of the Stories to next Sprint*".

Example by the numbers

Imagine a case where a SUT is covered by a unique requirement.
- If the requirement is in status **New** (this is a critical story that has been identified), its coverage would be 100%*__10%__=10%
- When the requirement becomes **Ack**, the coverage progresses and becomes 100%*__35%__=35%
- When the requirement gets **Approved** then its coverage becomes 100%*__100%__=100%

As you can see, 10%, 35% and 100% would be **customizable weight** we can affect to each status.

The same weight system can be used for each priority. Indeed, it makes sense to grant more weight to high priority requirements than low-priority ones.

Let's imagine you choose to use those weight settings:

Requirement Status	Weight
New	10%
Ack	35%
Approved	100%

Requirement Priority	Weight
High (P1)	100%
Normal (P2)	66%
Low (P3)	33%

Then, if we take our example with 10 requirements (where we're still missing 10, so the corrected requirement coverage is 50%), each having different status and priorities, we could have the following case:

Figure 4: Weighted requirement coverage

On the whole set of requirements we have, we get a global requirement coverage of 42%.
Here is how it is calculated:

$$\frac{100*(100+3x35) + 66*(5*35) + 33*(1*10)}{(4*100+5*66+1*33)} = 42\%$$

This means a weighted requirement coverage of 50%*42%=**21%**

Over time, the weighted coverage will progress, based on the number of requirements attached to the SUT and the status and priority of

each requirement.

Let's imagine we have a sprint with **14 user stories**. We do not expect more user stories so **the corrected requirement coverage is 100%**. All stories are at least in status **New**, you could get a progress profile like the one showed in **Figure 5**. In that example, one can see that at 50% of the sprint duration, we have reached 75% of weighted requirement coverage and that the number of requirements is still stable since the start of the sprint.

Not only can you manage your requirement coverage risk during the project, but you can also verify that it is covered 100% at the end of the sprint.

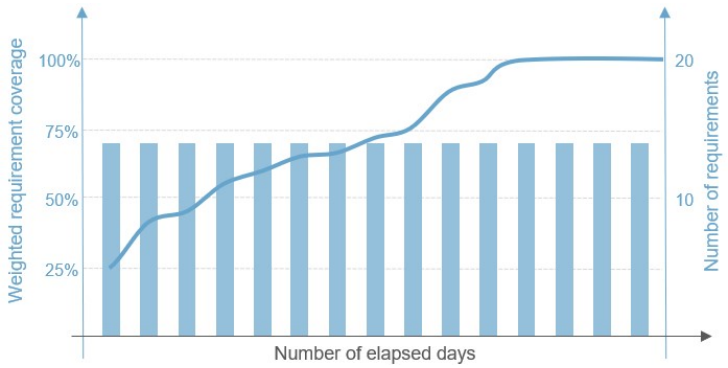

Figure 5: Constant number of requirements

Now, let us take a less favorable case. Imagine the product owner is late in defining the backlog. He/she proposes a limited number of user stories at the beginning of the Sprint, with the expectation that there will be more to propose during the sprint - well... we agree this is not a good practice when doing Scrum type agility. At the mid-sprint milestone, the situation is that only 40% of the scope is ready for implementation and some stories are still missing.

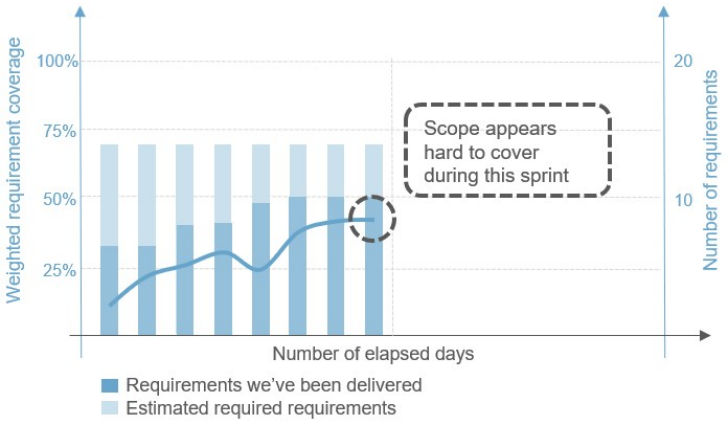

Figure 6: Late requirements definition

What would you decide in such a situation?
The practices you may have when facing such a risk (late scope definition and slow progress in defining them) are very much driven by the project context, and the organization culture.

Hence in this case, we could: - **consider reducing the scope as in Figure 7 (pushing back some user stories to the next Sprint) to get a chance to deliver something of value and with the right quality.** - Alternatively, you could **involve more resources** to address more stories... if you have such resources available in the mid-sprint (warning: this action has some limits, remember that 3 women cannot work together to give birth to a baby in only 3 month).

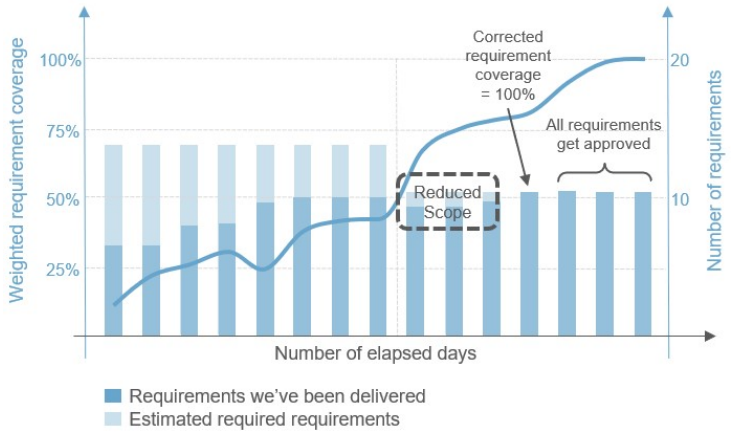

Figure 7: Reduced scope

Concluding on our example

Whatever your decisions are, given your constraints, what is important here is that you have a tool to measure and manage your risks regarding the scope. This is a Smart KPI. Any manager would greatly benefit from getting this KPI, in real-time.
XStudio calculates this for you, per SUT, per Folder of SUT... Hence this works for a product, an application, a Sprint, a Release or a Program.

No two organizations are alike. Therefore, the customizable weights we used in this paper are just examples. XStudio allows defining your own weights. The defaults value that XQual proposes are adapted for most organizations though.

Different organizations will be able to report and use this KPI for different layers or instance of governance: project manager, release manager, QA/UAT manager, CTO and even CEO levels.

Conclusion:

BECAUSE YOU GET THIS WEIGHTED SCOPE COVERAGE (WHICH CAN BE ALSO CALLED "READINESS") CALCULATED FOR ANY SUT OR GROUP OF IT, YOU CAN ASSESS YOUR RISKS ALL ALONG THE PROJECT/SPRINT. YOU DO NOT RELY ON "GUT FEEL" OR EXPECTATIONS AND ARE NOW ABLE TO MAKE DOCUMENTED DECISIONS.
IN THE SAME WAY, WE'LL SEE IN THE NEXT WHITE-PAPERS THAT WE CAN ACHIEVE THE SAME LEVEL OF INFORMATION, DRIVING TO SMARTER DECISIONS WHEN IT COMES TO SPECIFICATION COVERAGE, TESTABILITY AND QUALITY.

References:

(1) SWAG: https://en.wikipedia.org/wiki/Scientific_wild-ass_guess
(2) KPI: https://en.wikipedia.org/wiki/KPI
(3) Here we refer to the 3C: Card, Conversation, Confirmation

Figures:

TRY IT NOW!