Business rules in scrum projects

> *… a golden marriage or a divorce battle?*

The popularity of agile methodologies for software development is deserved. The predominant method of choice is Scrum. Customers may change their minds … but how does Scrum deal with policy makers that change their minds?

I have seen many rule authors (also business analysts or subject matter experts) struggle to deliver value in projects that use Scrum.

What is the issue?

- Business rules are rewritten into a user story introducing double work from the view of the rules author.
- The user story for a business rule is used in planning poker. The resulting plan consists of an incomplete set of rules causing the programmer to ‘creatively fill in the gaps’.
- The rules authors don’t keep pace with the development team. The development team continues and the rules are harvested from the code after the fact.

The resulting inefficiency or incorrectness is easily blamed on the rule author. Being a subject matter expert, he often does not have enough insight in Scrum for a good defense.

Scrum's popularity is due to faster software delivery with better results. How come Scrum teams struggle with business rules while they intend to have a flexible development method that is able to deal with changes? A logical reason may be that Scrum is just not designed to deal with business rules. Scrum is designed for problems that cannot be fully understood or defined. On the contrary, business rules operate in situations that can be fully understood and described, are always explicit and should be unambiguous.

> *That being said, the Scrum team still needs business knowledge and business rules.*

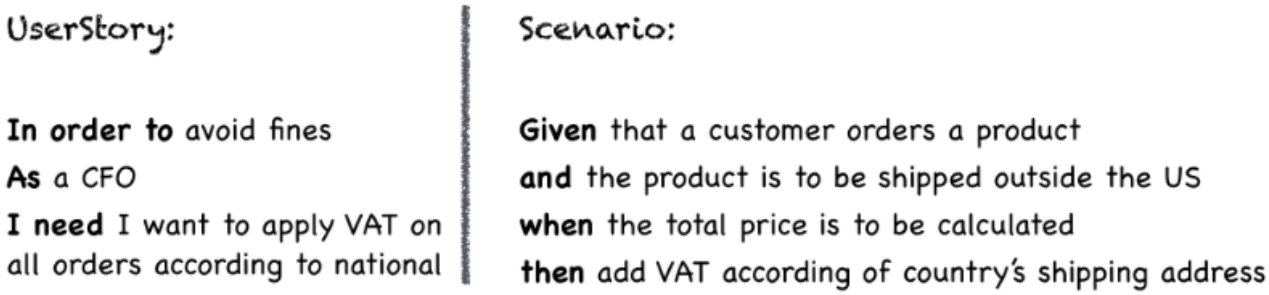
The business rules may be fully understood and described … but other aspects of a software product may not be. One way to deal with the situation is to deliver all rules in advance as a complete and consistent set.

This is undesired because:

- it feels like going back to the waterfall method that delivered such bad results,
- the project needs to wait for the rules, resulting in undesired delays,
- policies may still change during the course of the project.

Example

Let's go into a little more detail. A user story plays a central role in Scrum to scope, plan and evaluate a development cycle of three weeks.



No programmer can implement this user story without the need for extra information. In an attempt to provide complete information to the programmer it is tempting to associate the user story with a decision and the scenarios with rules.

UserStory:	Scenario:
In order to determine VAT **As** a CFO **I need** the VAT rules for each country	**Given** Shipping address in the Netherlands **and** the product is electronics **when** Product type and shipping address are known **then** VAT = 21%

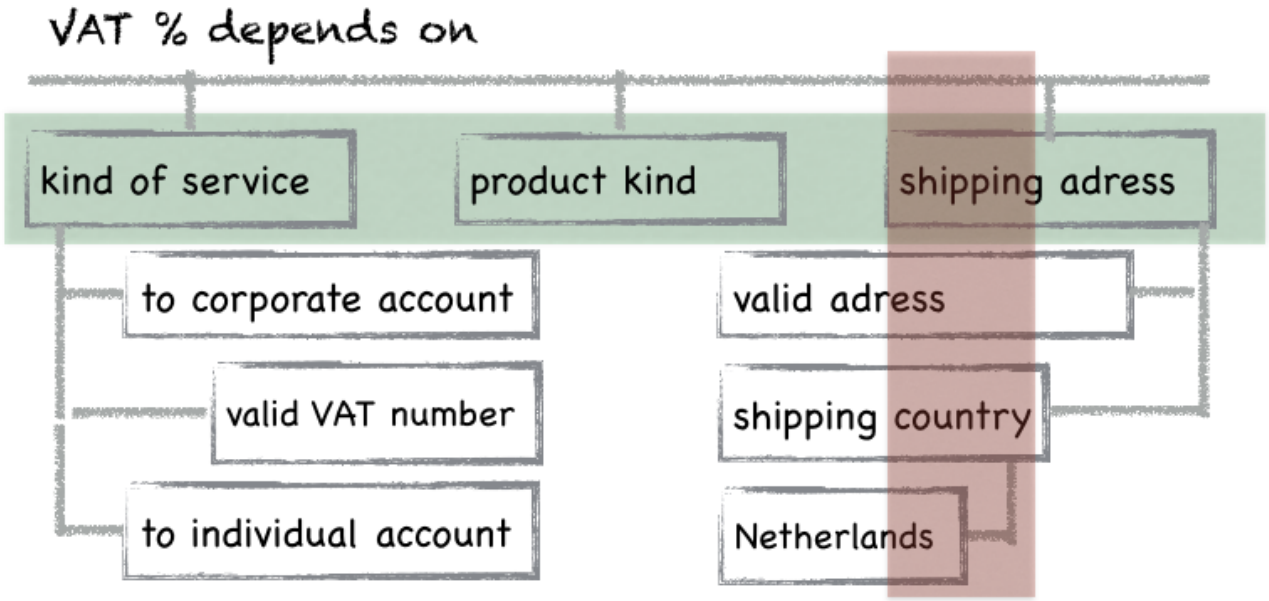
Other scenarios would need to deal with VAT for other kinds of products and other countries.

What we forget in these attempts is that user stories should be limited in detail by 'what can be hand-written on a small paper note-card'. Obviously all VAT rules of all countries can't be written on a small paper note-card. Rules should be considered ' an artifact' or 'data' in Scrum terms.

> *A happy marriage is the union of two good forgivers.*

The subject matter expert should help the Scrum team to define a good scope; the Scrum team should be aware of the special value of business rules.

The dilemma is that the Scrum team needs to cover all aspects of the software system in one Scrum cycle: Database, business logic, user interface, workflow etc. Therefore it is likely he tends to take a depth-first slice in a hierarchical knowledge structure: ship only to the Netherlands.



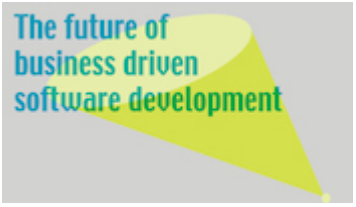
By doing so the resulting knowledge is always incomplete. One may argue that this incompleteness is inherent and will be corrected in future Scrums. True, but it means constant changes to earlier developed rules in every Scrum cycle.

> *A better way is to take a breadth-first slice in the hierarchical knowledge structure.*

The result is that details are gradually added and the delivered rules will be more stable.

Not all knowledge has a hierarchical structure. So here is my advice for the general case:

- The rules author may only deliver rules by a rules set that is well-scoped, complete and consistent. He should advise the product owner to choose a well-scoped user story.
- The scrum team must consider to add a user story such that a new rule should be like minutes' work and not hours' work (or days' work). Rule changes will be part of the Scrum life cycle, whether they are caused by policy changes, new insights or something overseen in earlier Scrum iterations.



This is just one step towards business driven software development. Looking forward to the next? Let me know by sharing this post.