Why SysML does often fail – and possible solutions

Nikolaus Regnat[1]

Extended Abstract: Did you try to bring model based development into your organization and had a hard time? Did you work hard preparing the process, methods and tools but despite all trainings and support, users did not adopt your model based development approach as expected? Did you experience low model quality and keep struggling to provide the needed model quality assurance? Well, you are probably not alone. Let us use the SysML as a prominent example to describe why modeling languages often do not live up to their expectations and why introducing them into an organization frequently fails.

It typically starts with a system architect who read or heard about model-based approaches and SysML. She/he may start talking with some of her/his colleagues and together they may convince their management that they should get rid of their handwritten, often huge and inconsistent, system architecture documents and replace them by a model-based approach. System architects are typically domain experts but very rarely modeling experts so they start to research the topic further. There are countless books regarding model-based development in general and the SysML in particular. There are numerous trainings available; big companies like Siemens may even offer in-house trainings. Let's assume that the system architects will get such trainings and have at least time to skim over one or two books. The best outcome they can get is that they now know a little bit more regarding the SysML language and realize that they would have to define how to apply the SysML in their organization. In other words: they know they need to develop a method that deals with their particular needs and describes how to apply the SysML in their specific use case. There are several possible scenarios that might follow; unfortunately most of them will often lead to failure.

The first scenario would be that the system architects try to introduce the SysML into their organization by themselves. This is a typical situation when the management is hesitant to invest larger amounts of money or the system architect is convinced (after reading those books and attending trainings) that he can do it by her/himself. Based on our experience this will not turn out right most of the time. While the definition of a method how to apply SysML is a hard task in itself one has also to consider countless other things: What is the purpose and goal of the modeling approach? Who are the stakeholders that directly or indirectly have impact on the approach? What are the skills of the potential users? How to train these users? How does the approach fit to the existing organizational structure and processes? How does the approach integrate into the existing tool landscape? Failure to consider these (and many other) things will typically lead to a modeling approach that does not fulfill all requirements of the

[1] Siemens AG, Corporate Technology, Research in Digitalization and Automation, Architecture Definition and Management, Otto-Hahn-Ring 6, 81739 München, nikolaus.regnat@siemens.com

organization and therefore is destined to fail.

The second scenario would be that the system architects realize that they need help to accomplish the task. If the management can be convinced to invest more money that typically means some consultants are hired. If these consultants are worth their money they will surely think about the above topics and define a modeling approach that fulfills all requirements of the organization. Unfortunately that does not mean that the introduction is successful in the long term. The worst situation is often when the consultants are also hired to do the creation of the models themselves. This is unfortunately still happening far too often, typically when the system architects are busy with their daily project work and the management thinks it is a good idea to “help” them this way. Not only will this successfully prevent any knowledge building within the organization itself, it will also often lead to a low acceptance of the approach: it often leads to the not-invented-here-syndrome of the organization. And when the contract with the consultants is not longer extended this usually means that there is no one taking care of the models any longer and the whole approach is soon to be ignored and forgotten.

The third scenario would be that either the system architects (first scenario) or the hired consultants (second scenario) did such an awesome job that all organizational requirements have been considered and a feasible modeling approach has been defined. Surely this will lead to a huge success and lead to another example of a successful introduction of a model based approach into an organization? Unfortunately, more often than not this is still not the case. Let’s go through the three most important reasons why the introduction of a model-based approach into an organization might still fail.

Lack of management support is one of the main reasons. Without the management actively contributing to the success of the approach failure is often inevitable. People need to be both motivated (in regards to their goals; e.g. “we want to replace our hand-written documents within the next fiscal year”) as well as given enough time to get accustomed to the new way of working. However, more often than not system architects do not get enough time to actively work with the modeling approach as their day-to-day business is taken priority. Moreover, the model-based approach has to be constantly tailored and improved to suit the changing needs of the organization and the management has to actively support this too.

Lack of proper model quality assurance is another main reason. It typically does not cause short-term issues but will often lead to failure in the long-term. Unfortunately many organizations realize this too late: models that are not constantly monitored using both a combination of automated checks and manual reviews will massively degrade over time. It often results in something we call “model graveyards”: an unstructured collection of data, often with unreadable and unexpressive diagrams, unused model elements (deleted only from diagrams but not from the model) that is inconvenient to extend and hard to maintain. Fixing these issues will often cost so much time that it is not feasible to do it.

The third most prominent reason for failure unfortunately is the SysML and accompanying tools themselves. It is the prominent lack of focus on the end users. The SysML was built *by* modeling experts *for* modeling experts ignoring the fundamental fact that most end users are not and probably never will be modeling experts. To make things worse, the typical SysML tools were also built with a focus on modeling experts in mind. This combination of a general purpose language that lacks any method with a tool that is focused on expert users leads to a situation that is the downfall of many modeling approaches. Users will find themselves very often in a similar situation. They have gotten training on SysML and training on the method developed for their organization. However, most users will not use the modeling environment on a day-to-day basis and thus will only very rarely become modeling experts. They therefore struggle every single time they use the modeling language and accompanying tool. Users will typically have to go back and forth between guideline and tool to find out how to model, resulting in a massive productivity loss. Instead of being able to work on their original task (i.e. describing their system architecture) they struggle to handle the language and tool. This typically leads to frustrated users, lowering the acceptance of the modeling approach massively. Users will try to avoid using the modeling language and tool and find creative ways to circumvent the approach. Things that cannot be expressed properly in SysML are not improved in the modeling approach but instead other tools are used to draw “pretty” pictures. Existing models will get out of date and in the long run will not be used any more. This goes on until one got back to the point where all started: hand-written documents with some pictures or diagrams in it.

Knowing all that what can be done to improve the situation and successfully introduce a model-based development approach into an organization? Based on our experience regarding model based development in industry one has to focus on the three main aspects mentioned above. From day one ensure that the management supports the approach and make clear what is needed. Establish a model quality assurance process as soon as possible. And focus on the user experience. This cannot be stressed enough: even with proper management support and established quality assurance it is the acceptance of users that make or break the success of a model-based approach.

But what can be done immediately so that both modeling languages and tools improve on the situation? We think that it is the main responsibility when defining a modeling approach for an organization to improve the user experience as much as possible. The language has to focus on the user’s needs and do not provide unnecessary things. The tool has to support the defined method and guideline as much as possible. Every stumbling block for the users that can be removed has to be removed.

Some tool vendors have built their tools so that this can be already done in a sensible way. Unfortunately other tools only provide very limited capabilities to do so. As always: you have to choose the right tool for the right job. We are currently focusing on MagicDraw, as it supports a wide range of customizations that enhance the user experience. During the last years we have systematically worked on correcting both language and tool shortcomings. We’ve started with the easy things: we provided model templates for our users. These templates match both the structure defined in the guideline as well as the generated documents as far as possible. We then used the tool customization possibilities of MagicDraw to guide the user wherever possible: instead of allowing to create every element and diagram everywhere the context menu only allow to do the things that are sensible in that context. We did not force users to remember (or read in the guideline) what additional stereotypes they have to add to various SysML elements. Instead we created new element as well as diagram types for them. We also worked on improving the visualization of model information. Elements may change their color, show icons etc. based on properties of the model. We even let users setup these things in an easy and consistent way: the user can define a connection type (e.g. CAN Bus) by himself and give that new type a distinct color. The tool will take care of this and ensure that both icons and diagram elements show this color whenever the type is used. Did you ever model a SysML Block with 10 ports an instantiate that block? Depending on the used tool the user might end up with a cluster of ports that one has to manually arrange again, every time an instance is created. This is just an awfully bad user experience, and SysML tools did it this way since they have been made available. Using the tool API we've implemented a very small extension that allows defining the port layout of elements once and whenever an instance of such an element is created the ports show up as defined. It is such a small thing but modeling experts and tool vendors alike ignored it and nobody told the tool vendors to change this (well, we did and MagicDraw now has such an option built in). When putting a strong focus on the user experience one can easily identify dozens of things that should have been improved a long time ago. And it is often a combination of small things that lead to a bad user experience.

The attentive reader might have already guessed it: ultimately we went the long road of implementing domain specific languages (DSLs) based on the UML/SysML using MagicDraw. We went even further and created a set of building blocks that allow us to efficiently create new DSLs with manageable effort. Our experiences in various projects over the last 5 years are more than promising: the user acceptance is very high and the overall model quality is far better at a reduced effort than anything we've done with SysML. All of this also helps to convince the management that the approach leads to an increased productivity and improved quality; in other words to a good return on invest.

Now, we don't propagate that everyone should go this route; instead we want to show that with a strong focus on the end users we could have been much further regarding the widespread acceptance of modeling languages and accompanying tools. Our approach shows what is already possible when customizing and adapting both languages and tools. What could we do if languages and tools would have been better at this? To re-enforce my former statement: both the modeling community and tool vendors need to step back and re-think what has been done in the past. It is not the goal to train domain experts to be modeling, language and tool experts just to be able to do something meaningful. The goal is to enable domain experts to focus on their challenges by providing modeling languages and tools that empower them to do so. Without us focusing on the user experience the introduction of modeling approaches into organization will continue to fail.