The f-word in systems engineering

oose.

The pitfalls of choosing the wrong modeling element for your modeling purpose

Axel Scheithauer
oose Innovative Informatik eG, Hamburg
SWISSED – 18. September 2023

oose.

Axel Scheithauer

Trainer, Consultant and Coach

linkedin.com/in/axelscheithauer

xing.to/Axel_Scheithauer

@AxelScheithauer

Model Based Software
and Systems Engineering
Business process modeling

OMG Object Management Group

UNIFIED MODELING LANGUAGE

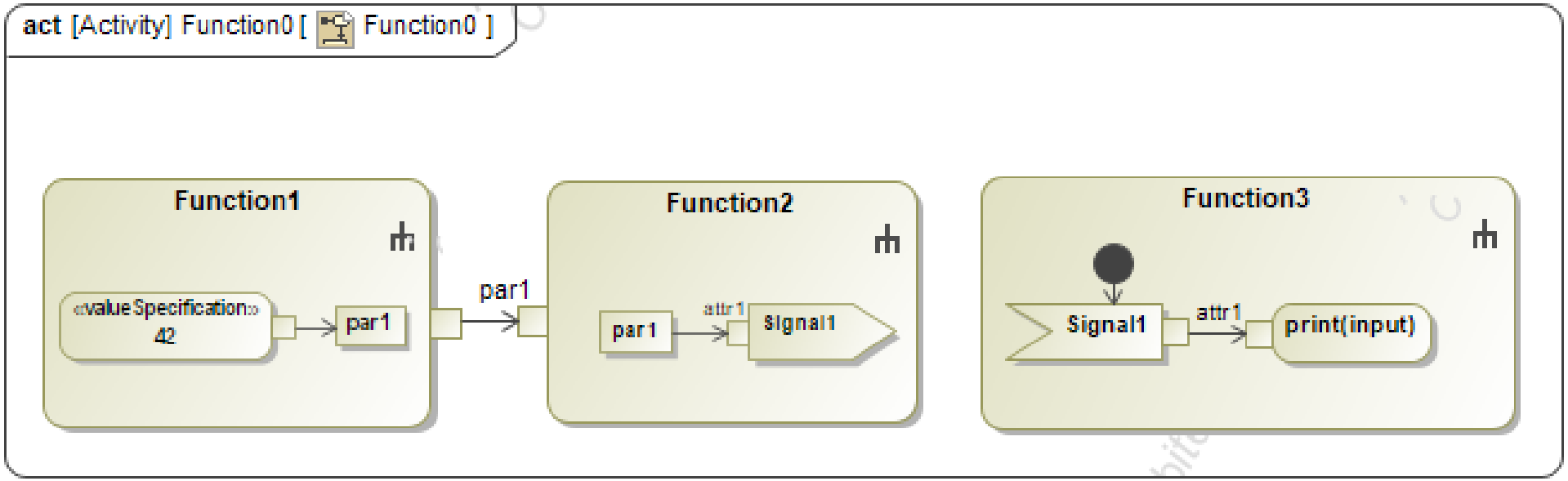
OMG SYSTEMS MODELING LANGUAGE

oose.

Agenda

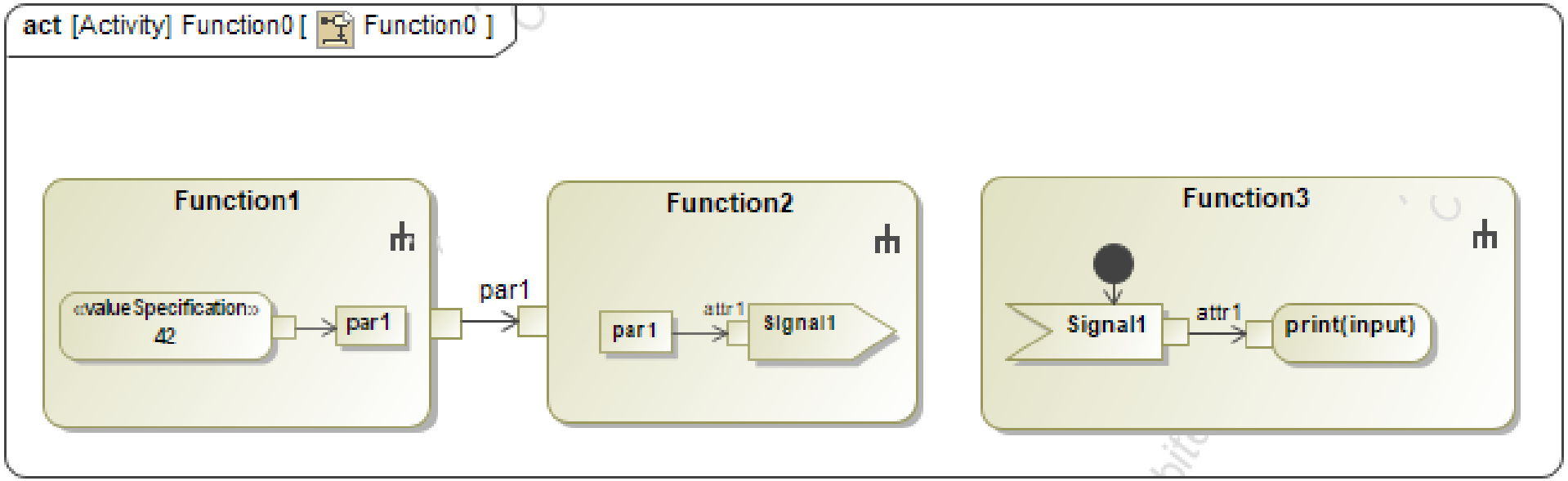
1. Functional Analysis
2. Allocating behavior to structure
3. Some problems
4. What it actually means
5. Shortcomings of activities for the *f-word*
6. My solution
7. My definition of the *f-word*
8. A real example
9. Conclusion

Result of a functional analysis of one use case



I tried to come up with the simplest example, I can think of: Three activities, where one has an object flow to another one, which sends a signal to the third.

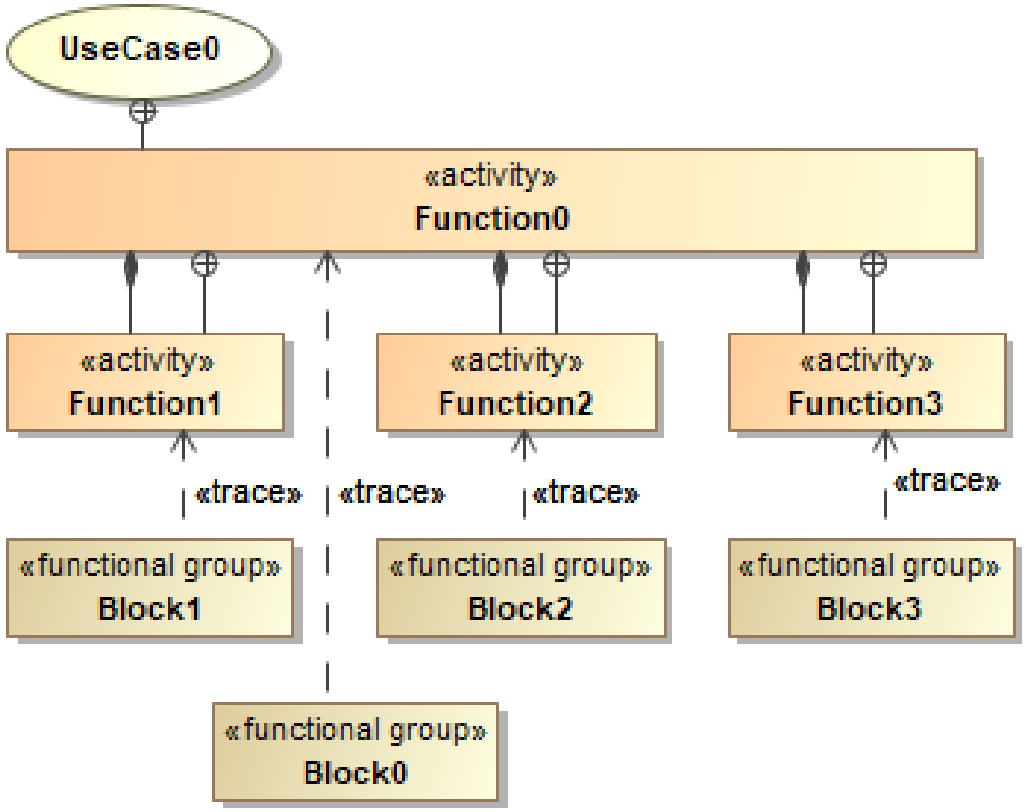
What are the semantics of this model?



We can simulate the model to explore the semantics (Demo)

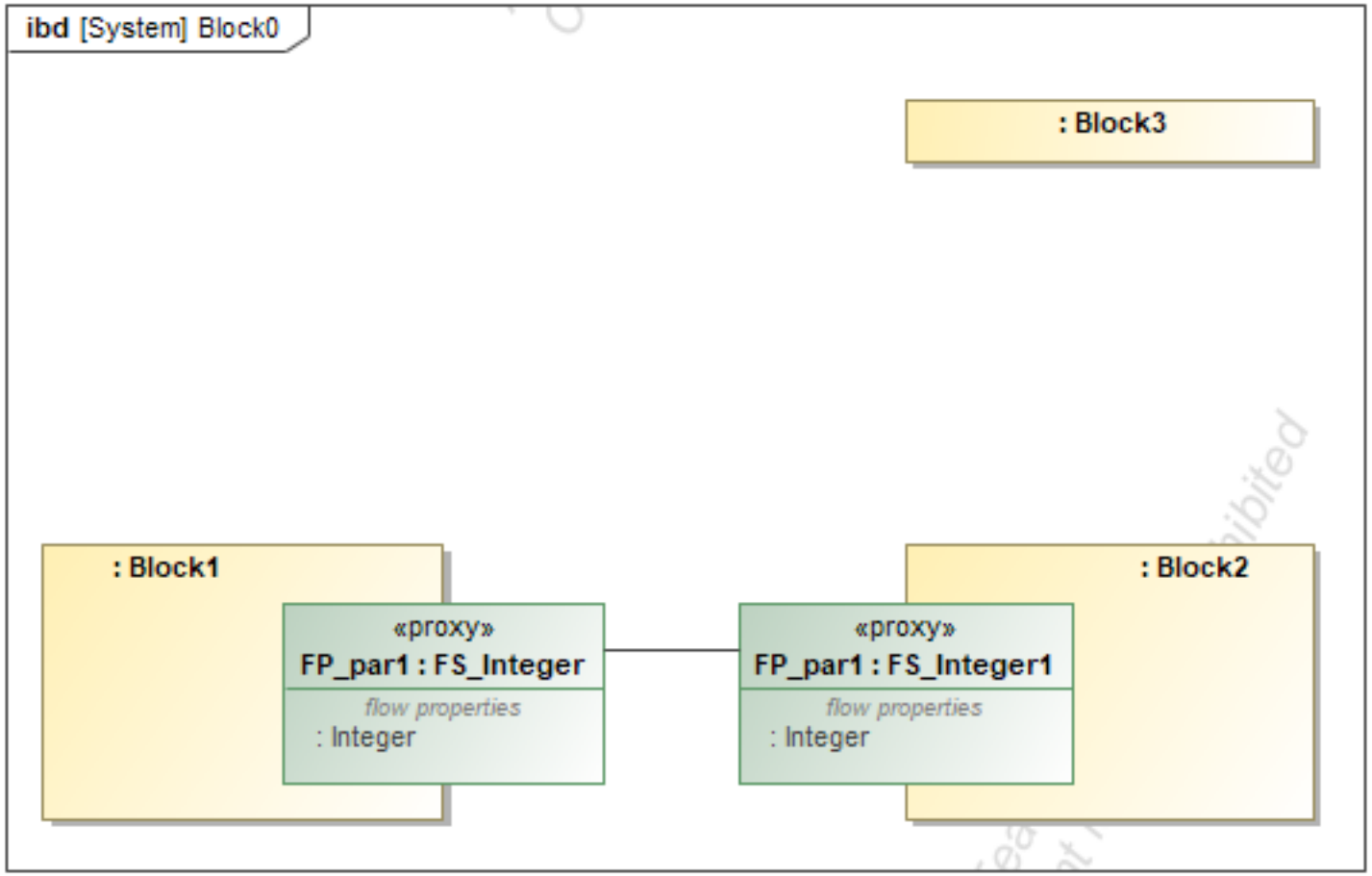
(we can also read the lengthy explanations in the SysML, UML and fUML-specifications)

Allocating behavior to structure



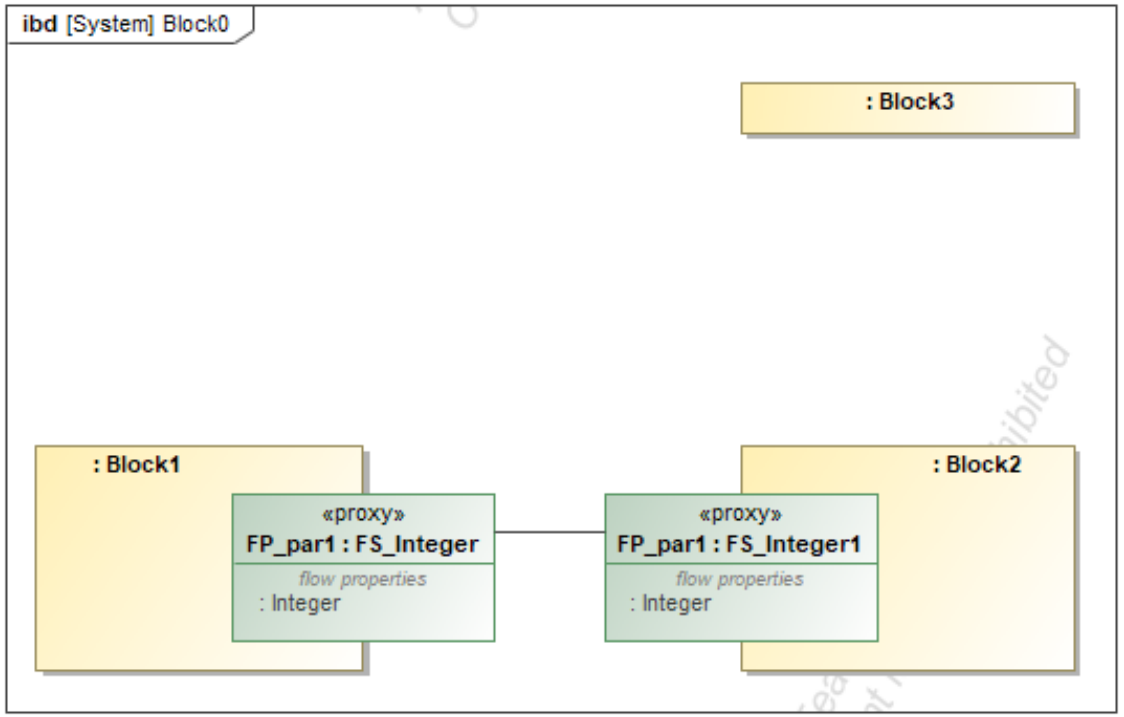
In the FAS-method, f-words are grouped and then assigned to functional blocks (not shown here).

Functional architecture



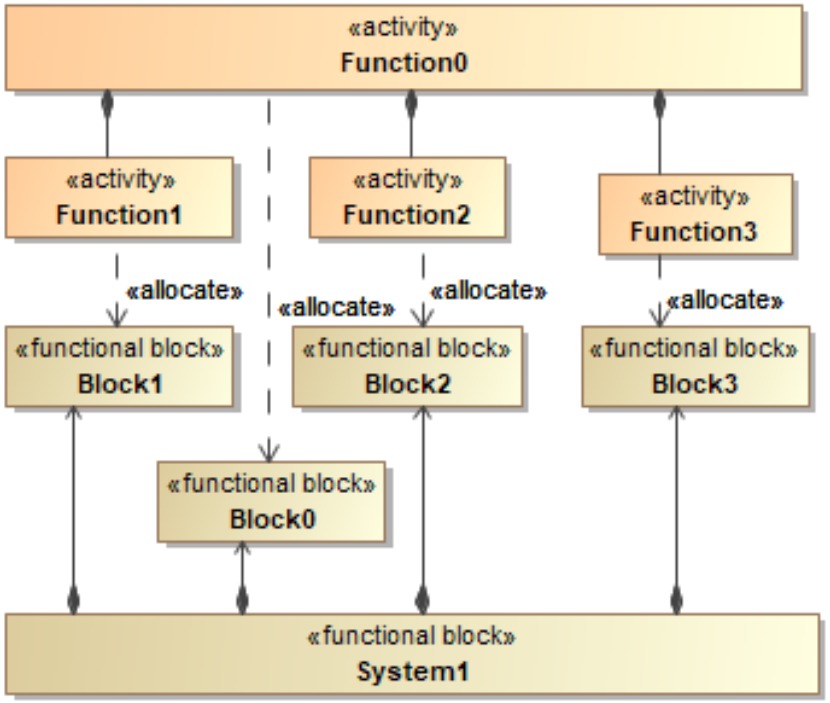
The FAS-plugin automatically creates a functional architecture.

Missing elements



Block0 is the system. It doesn't have an own behavior. The behavior assigned to it is the description of the use case. The interface between Block2 and Block3 is missing, because the plugin currently doesn't support it (it's a bit more complicated to evaluate).

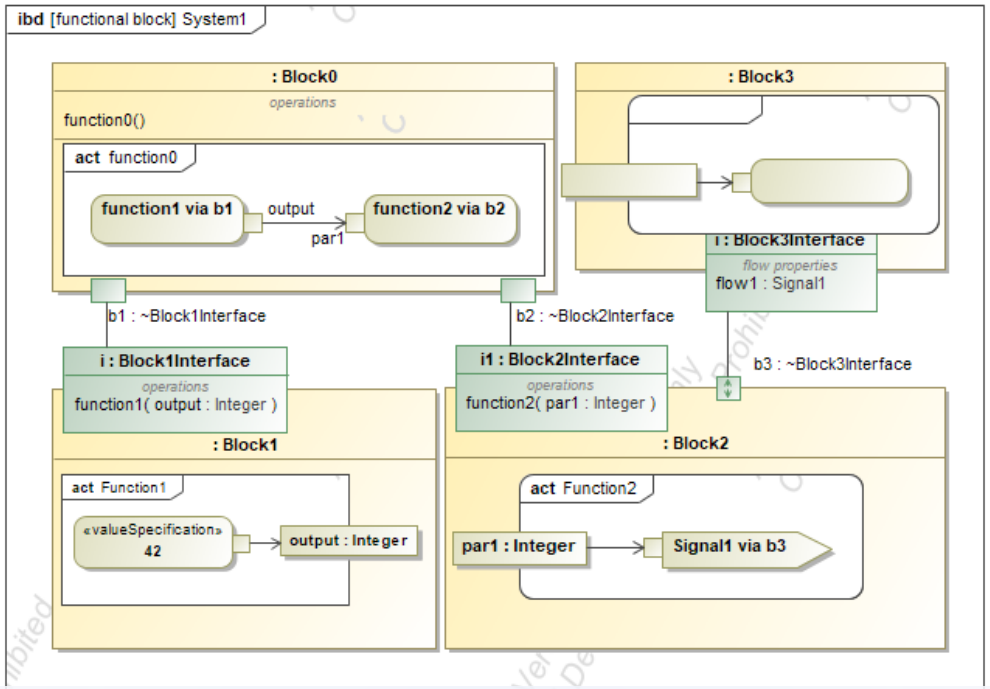
This leads to following model



All activities are behavior that needs to be allocated to some structure. All structure blocks together belong to a system.

Of course, a creative stretching of the SysML rules is completely fine. The point is, you must know the rules to break them.

A simulatable model of the architecture



All activities now have their home. I manually created the interface between Block2 and Block3. Block0 now has the responsibly to invoke behaviors on Block1 and Block2. This was not the interpretation in the first example.

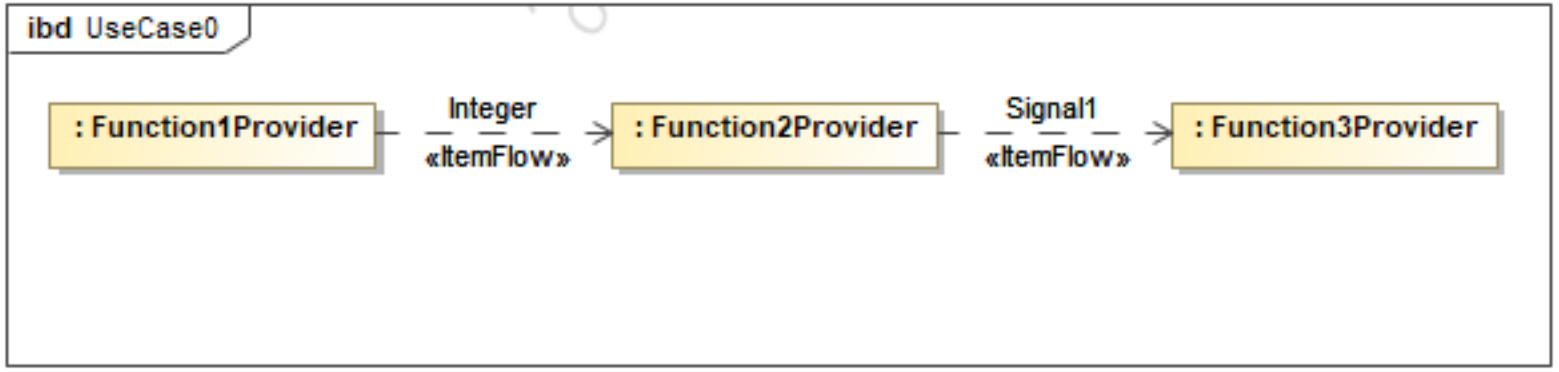
This can be simulated (Demo).

Shortcomings of activities for the f-word

Note: The following should not be understood as critique of the FAS method. After all, it is a quite successful method that has proven its worth. However, I see some weak points, and try improve on them.

- The role of the use case activity is unclear (at least to me)
- The activity diagrams define semantics, which are not needed for the functional architecture: Invocation, control flows and nodes.
  - Added complexity and work
- Activities are used for three incompatible and not clearly distinguished purposes: describing the steps of a use case, describing algorithms, specifying the flows between behaviors.

My solution



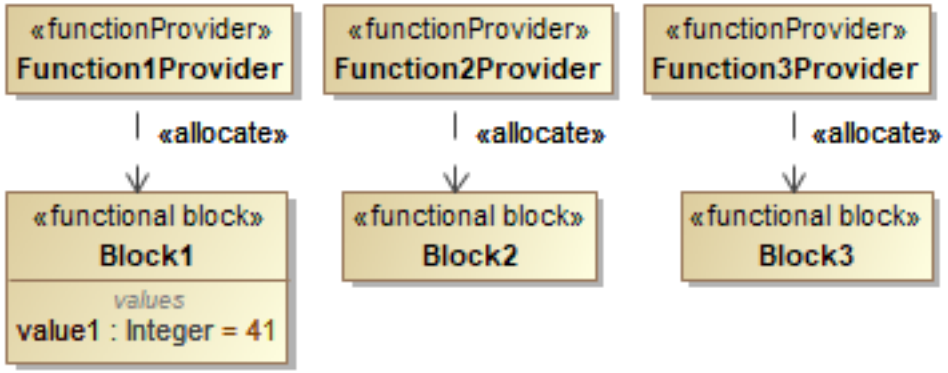
A use case analysis describes the function providers and the items flowing between them. Each function provider is responsible for exactly one activity.

This avoids the overspecification activity diagrams simply require. It is easy to create and it is easy to create a functional architecture from it.

This can not be simulated (maybe there is a way - I'm still experimenting)
** I also heard this concept being called "function enabler".*

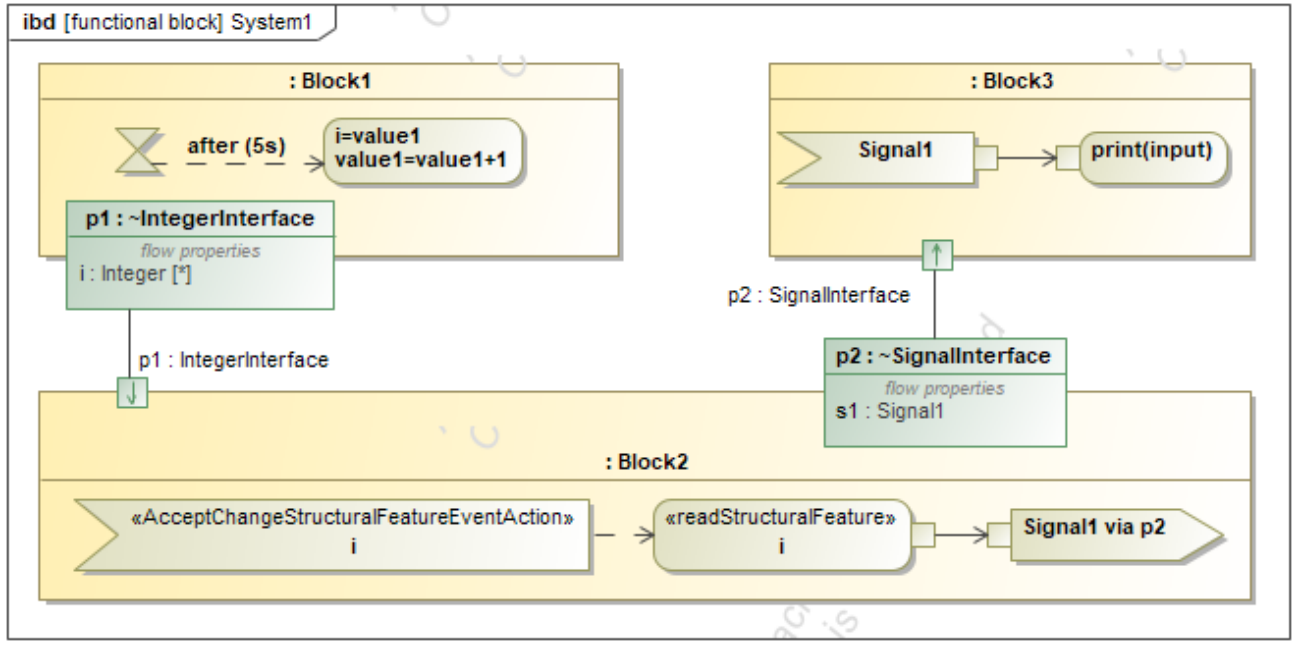
oose.

Allocating behavior to structure



As before, behavior needs to be allocated to structure. This time the behavior is hidden inside its host, the function provider.

The functional architecture

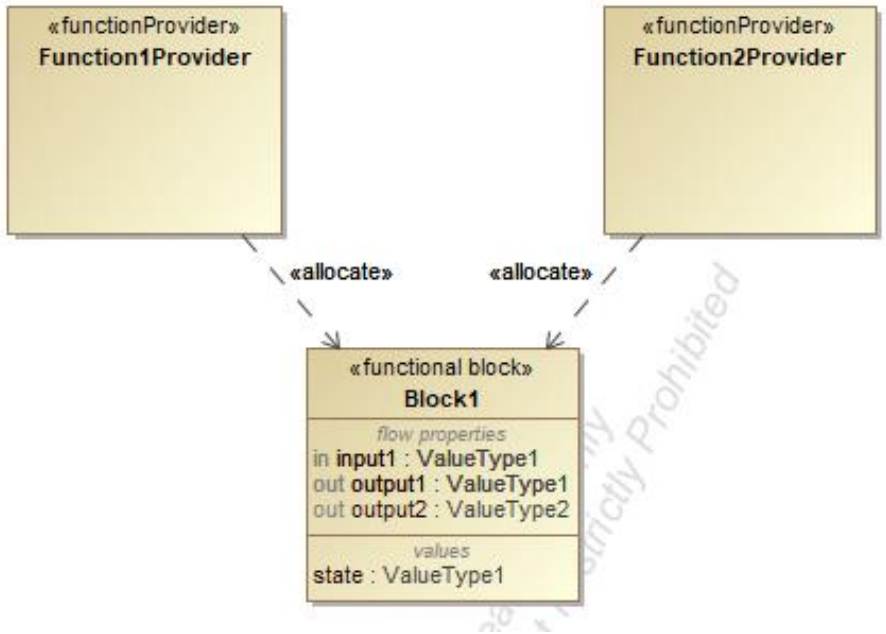


Block0 doesn't exist anymore, since no activity was used to describe the flows between behaviors.

I think, this is, what the original diagram was supposed to describe.

This can be simulated (demo)

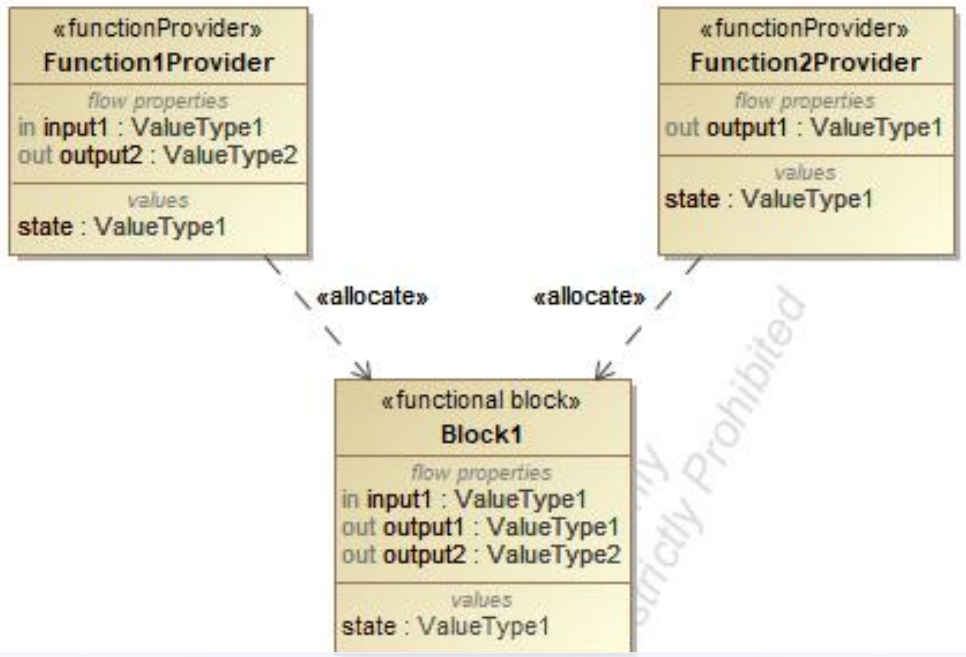
Combining function providers



In general, two or more function providers will get allocated to the same functional block. The FAS-method has some heuristics which ones should be combined.

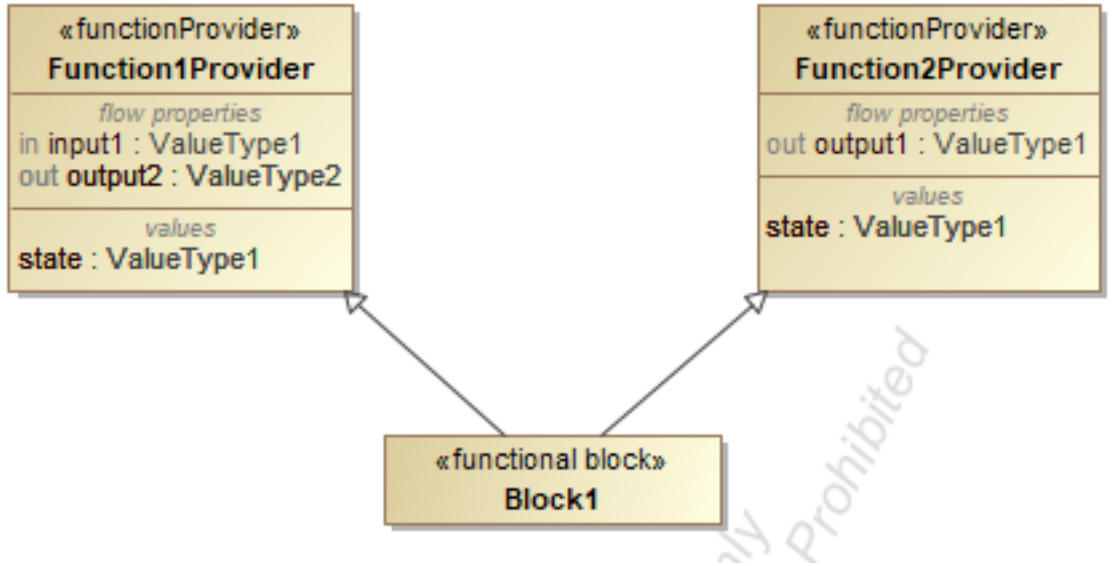
This functional block define the in- and outputs and system states it is responsible for.

Specifying function providers



Actually the in- and outputs are deduced from the function providers. So they could be modeled with these values in the first place. And with the state as well.

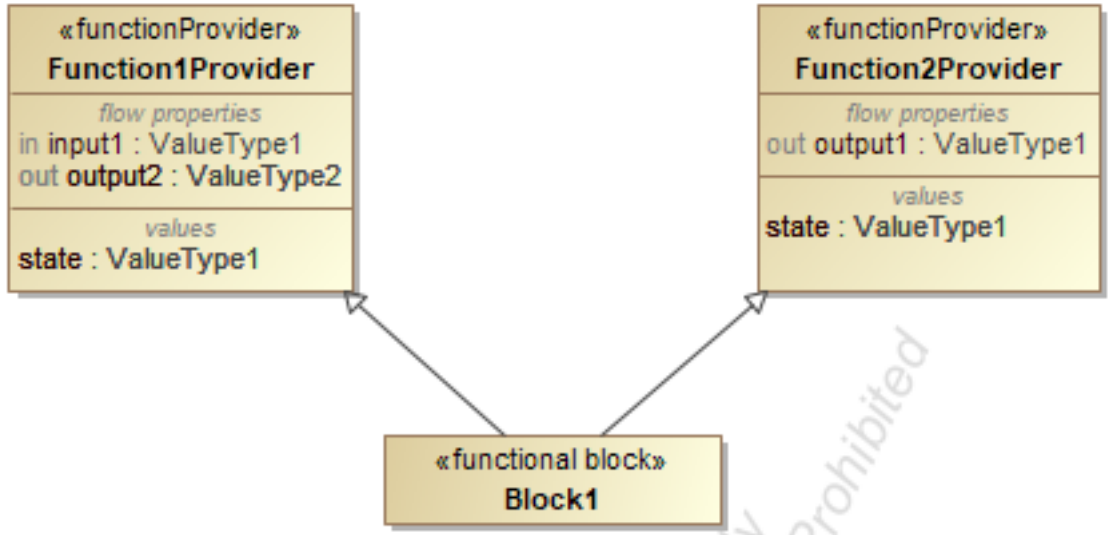
Specializing function providers



If the function providers already specify everything, we might as well the block inherit the properties from them.

Both function providers have the same state. This is a reliable heuristics for combining them.

My definition for the f-word



VDI 2221: A Function is a solution agnostic relationship between input, state and output of a system

Well, maybe the function provider is the long sought after formal definition for the f-word. Much to my surprise, I found that the VDI 2221 uses function with the same meaning here.

Note: I'm not saying, that this is the only correct definition and everybody else is mistaken. I think it is one, that is helpful for the FAS-method. Maybe also for other methods.

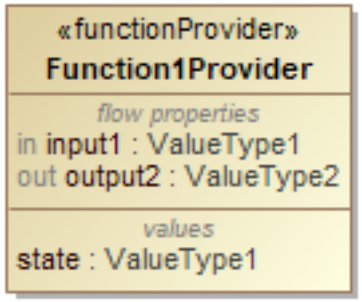
oose.

A real example

<tbd>

oose.

Conclusion



Axel.Scheithauer@oose.de

In MBSE you need to define a function formally with the help of a modeling language.

For the FAS-method I think the function provider is a useful definition.

Other methods might have other definitions.

It is important to carefully select a fitting modeling element. Otherwise, the model will contain accidental complexity.