On universality of bindings in AADL and resources concept of bindings

Denis Buzdalov, Alexey Khoroshilov
ISPRAS

September 29, 2016

1 Introduction

One of the reasons to create models of complex systems is analysis of these models. Modern complex safety-critical systems are obviously needed to be analyzed automatically or at least in an automated way. So, modelling languages should provide ability to do this and should try to make this to be done conveniently and without much overworking.

One of consequences of this idea is that we want to perform analysis similarly for those model characteristics that in fact are similar (which probably is not obvious when looking at a glance).

This allows

- modellers to use similar syntax when describing different similar properties in models;
- both modellers and analyzers to reuse same well researched concepts for different characteristics for free (we will show examples further);
- analyzers to have a single instrument for semantically different but (in fact) similar ways of analysis.

So, what we want (simplified) is to reduce effort of people who are writing models, who are applying analysis instruments and who are creating these instruments. This is suggested to be done with rethinking and generalization of some basic concepts of AADL.

2 Problem

At the moment AADL supports different types of characteristics some of which are looking very similar and some of which are modelled in a significantly different way.

The standardized way of non-direct links in the model in AADL is binding. At AADL of version 2.2 there are four types of them: processor, connection, memory and functional bindings. At the moment they are set with standard properties but it is decided to use special syntax constructs in further versions.

Other relations between model components (e.g., power consumption) are not standardized and appropriate analyzers need to work with their own property sets or annexes to be able to add appropriate information to model. The inconvenience in this case is that when using such an instrument-oriented property set, an AADL model becomes not interoperable and can be analyzed (for this characteristic) only with a single analyzer.

This situation repeats each time when users want to model different relations between model parts that are not in the standard already.

So, there is a problem that modern AADL needs not provide *extensible* and *flexible* way to represent *binding-like relations* between model components, including both standard and user-defined ones.

This is a widely discussed theme in the community and there are some ideas how to manage with this. We want to present our thoughts on this problem and to state them in the written form. This is considered as a point for the future discussion.

3 Proposition and discussion

3.1 General idea

The ways we think to deal with the problem is to consider notion of providing and receiving of various *resources*. *Resource* is a something that is required for receiving components to perform what they are targeted to. Other components can provide *resources*, in most cases with some parameters and limitations.

Some of types of resources are measurable, some are not. We will discuss some examples of both below.

Obviously, in this approach some components can simultaneously be receivers and providers of particular *resources*.

3.1.1 Arity and naming

It is important to understand that we see providing and receiving of resources through named and typed *points* (similar to *binding points* notion), i.e. components can define several *resource providing points* and *resource receiving points*.

When there is a finite set of resource points each having its unique semantic meaning, each one should have its own name and should be connected individually. When there are several resources points with the same meaning, they can be treated as an *array of resource providing/receiving points*.

> *For example, we can consider a* multiplexor *providing several downstream* `connectivity` *resources and requiring an upstream* `connectivity` *resource of appropriate bandwidth.*
>
> *Depending on whether this multiplexor has difference between downstream connectivities and depending on the level of abstraction, these downstream providing points can be modelled though several named points or as an array of downstream points.*
>
> *For example, if such multiplexor is rather a simple thing which multiplexes messages of constant sizes of 8 and 24 bits respectively at the same rate and producing 32-bit messages, it can be modelled as a component with the following resource points:*
>
> - *a single receiving* `connectivity` *resource point named* `upstream` *capable to send 32-bit messages;*
> - *two* `connectivity` *resource providing points with names* `downstream_8` *and* `downstream_24` *with appropriate limitations.*

If we do not want to see such details in a model or if multiplexor does some work on distinguishing between downstream connections, its `connectivity` *resource providing can be modelled as an array of resource providing points named* `downstream`.

3.1.2 Interesting particular cases

In case when a component receives one type of resource and provides another one, it can be named *resource converter* or *resource adaptor*.

We can consider a case when a component receives and provides the same *type of resource*. Such component can be called *resource allocator* or *resource distributor* because it can have different limitation at the receiving and producing sides.

Multiplexor from example above is a nice example of resource distributor.

Upstream and downstream resource points have different requirements on bandwidth because (in the simple case) upstream bandwidth has to be not smaller than sum of bandwidth of downstream points.

3.2 Examples from AADL core

We can consider examples of *processor power* (which is a generalization of *processor time*) and *connectivity* resource types. At a glance, relation between receiver and provider of the *processor power* resource type looks similar to current AADL's PROCESSOR BINDING of receiver to provider. This looks to be measurable, but the measurement unit is obscure.

Connectivity resource type and CONNECTION BINDING have the similar relation.

But when we look deeper, we find out that PROCESSOR BINDING has at least one more semantics which can be treated as one more *resource type*. By definition, PROCESSOR BINDING means also that bound component is scheduled by those it is bound to. Actually, *schedulability* can be also treated as one of *resource types*.

And, by the way, in case when PROCESSOR component represents both processor hardware and operating system, then this component provides both *processor power* and *schedulability* resources. When at some point of model refinement or when modelling a multicore system running a single operating system, hardware processor and operating system can be modelled with separate AADL components. Thus, in this case different AADL components — hardware processor and operating system — would provide only *processor power* and *schedulability* resources, respectively.

3.3 Cases out of AADL core scope

Some of models characteristics looks very natural to the resources approach. Models can be analyzed with same methods and instruments for these characteristics if they were defined in the same manner. At the moment people have to define their own property sets and implement their own analyzers for such cases.

Very good looking example is *electric power* consumption. It seems to be very natural to treat it as a *resource type* and to analyze, for example, sufficiency of this resource in the whole model. It looks like the same methods can be used either for sufficiency of *electric power* and *processor power*, and for *connectivity* bandwidth. These

resource types have similar mathematical basis: they are additive with possible overhead in each *resource converter*.

This additive basis implies one more property of such resource types: if all values are positive, path though providers/receivers have to be acyclic. Possibly, this can be applied not in every case but still this check can exist in analyzers. And because of the same nature, this check can be implemented in the same way for all these tree types of resources.

Looking at these resources as similar things would allow us to reuse analyzers and to make writing models to be easier.

> *By the way, even on additive resource types, sufficiency of particular resource does not mean actual ability for connection of producers and receivers.*
>
> *For example, having powerful enough provider of* `electric power` *resource, it does not mean that we can connect it to some particular receiver. As it was said before, such connection may have additional properties. For example, electricity provider and receiver must be compatible on voltage and on whether current is alternating; for alternating current, frequency must be compatible too.*
>
> *Thus, some components in models (like voltage transformers) may receive and produce the same type of resource of similar value (possibly, differing by conversion overhead) with different parameters (like voltage for* `electric power` *resource).*
>
> *Additivity sometimes can be conditional (or parameterized). For example, light (which can also be treated as a resource type) and alternating current have phases. Phases dramatically influence on the value of summing of original resources. That is, summing of two* `light power` *resources with luminosities l_1 and l_2 respectively, depending on phases we can get any value between 0 and $l_1 + l_2$.*
>
> *So, idea is that parameters of resource types should be formalized somehow for detailed modelling. At some level of abstraction, such details may have no meaning yet.*

So, the idea is that users can define their own resource types which are not provided by the standard and use these user-defined resource types in a similar way using the same analyzers. This looks like a great feature of nice modern modelling language.

The idea can be generalized using some notion of *resource category* incorporating different *resource types* with similar properties that can be analyzed in the similar way. Those three resource types can be united by *additive measurable* resources category.

> *Probably, it would be useful to consider resource category of* `decision taking` *or general* `functionality` *resources. They are not measurable. Connection between such resource provider and receiver means ability for provider to* influence *on behaviour of receiver or even to* perform duties *for receiver, to have functionality being delegated to them.*
>
> *In fact, it seems that considered above* `schedulability` *resource type should belong to this category.*
>
> *Also, human interface devices can require* `decision taking` *or other* `functionality` *resource from people. This allows in models to distinguish devices that decide something in their own and devices that just receive and transform decisions taken by human.*
>
> *The same ideas can be considered when some subsystem is actually a proxy of another one. Such proxy requires* `functionality` *resource provided by one that*

actually does the work. Human in the previous example is such a provider, but any electronic subsystems can be considered similarly.

3.4 Reliability or quality of resources

Very important notion that raises when looking even at standard bindings as resources is *reliability* or *quality*.

3.4.1 Connectivity and duplication example

For example, in connectivity case we can consider each single CONNECTION BINDING as providing two resources: `connectivity` resource itself and somehow measurable `reliability` resource.

With this division we can easily model two patterns of duplication. Consider a duplicating component requiring two `connectivity` resources of bandwidth b_1 and b_2, and two `reliability` resources of values r_1 and r_2 respectively. For speeding up, this component provides (in the example case) `connectivity` resource of bandwidth $b_1 + b_2$ and `reliability` resource of value $\min(r_1, r_2)$. For raising of reliability, the component provides `connectivity` of bandwidth $\min(b_1, b_2)$ and `reliability` of $r_1 + r_2$.

Generally, `reliability` considered as a resource has min-based mathematics basis, unlike sum-based `processor power` or `electric power`. It means that usually aggregation function of two values of this resource type is taking minimum instead of taking sum. This fact is important for potential analyzers. Other resource types that have the same property can be included to min-based resource category, which can have same analyzer and algorithms of management.

3.4.2 Issues and discussion

There are some issues and open questions on concept of reliability as a resource. It is a very debatable question whether or not `reliability` is

- a resource itself or
- an additional characteristic of resource providing or
- a characteristic of a component itself.

It seems strange for a component to provide `reliability` resource as the only provided resource; the same about receiving. Also, it looks hard to understand what it means when some component provides several pieces of `reliability` resource having, for example, a single `connectivity` resource. Miswiring of several pairs of `connectivity` and `reliability` is also a big issue.

Another question is whether reliability can be applied to all types of resources or not. It seems that some resources like discussed above `schedulability` and other `decision taking` resources implies full reliability thus not requiring to define and to measure it.

Again, unit of measure of reliability (whether it is treated as resource or an additional characteristic) is a point of discuss. It seems that different resources and resource types (if applicable) may have different units of measure, like *count of failures on time* or *count of failures on data amount* for `connectivity`, *dispersion of frequency* for `electric power` and etc.

In our opinion, considering *reliability* as a characteristic of resource providing relations of some types is the best alternative between those above. But its calculation and

derivation would definitely use notion of *reliability of a component* itself which seems to be not a resource but a value.

Also, different variants of such reliability can raise. For example, we can consider *reliability* of the whole array of resource points (or of a point which can provide resource to multiple receivers) which is *potential* or *maximal reliability*. We can also consider *reliability* of each array element (or of the 'multiple' point considered with some particular resource receiver) which is somewhat *actual reliability*. So, idea is that reliability is not so simple characteristic.

4 Conclusion

We put special attention to the problem of universality and extensibility of binding notion. Some significant relations between components in a model look like bindings and should be treated similarly.

The current idea for newer version of AADL is to consider bindings not as a usual property of a component but as a special construct (both semantically and syntactically).

What we suggest is to make this construct extensible and flexible. To reach it, we present a notion of *resources* and propose to treat standard bindings as a relation between resource *provider* and resource *receiver*.

We think such relations should be *typed* and standard variants of bindings should be variants of such types. Users should be allowed to use their own types of resources because no standardized list of such types would cover all variety of resources that can be used in modern models.

Since these types have similarities in some points, it is suggested to group types of resources to *resource categories*. It allows to reuse modelling techniques, analyzers and algorithms for analysis of models with characteristics related to different resource types that belong to same resource categories. Generally this would simplify both processes of creation and analysis of models.